Hunt for Unused Servers

Nikolai Joukov and Vladislav Shorokhov
modelizeIT Inc.

Abstract

Modern enterprises critically depend on IT. However, enterprise IT environments are complex and poorly documented. As a result, various IT components are forgotten and unused. Recent estimates show that 30% of servers in datacenters are unused on average. Elimination of unused servers (physical and virtual) and software instances decreases costs, electricity consumption, risks of problems causing business interruptions, and security exposures.

IT components form complex interdependent graphs. It is intuitive to declare the nodes not used by other nodes to be unused. However, the reality is a lot more complex: servers (even if unused) are highly inter-connected.

This paper has two main contributions. 1) We present a practical method to detect unused servers based on the dependencies graphs. The method relies on the dependencies classification and propagation of usage information along the graph. 2) We apply and evaluate the topological method and utilization-based approaches for real enterprise datacenters. We benchmark and compare both methods in terms of detection error rates.

I. INTRODUCTION

Enterprise IT is complex and constantly and rapidly changing environment. Banks, insurance, retailing, manufacturing, and most other companies have heterogeneous IT environments where each business application is implemented on a few servers. This is dramatically different from the IT environments of some large IT companies where a large pool of servers performs the same function (e.g., [14]). For heterogeneous IT environments, documentation never catches up with reality and can never reflect all the details and complexities. Each software component can have from dozens to hundreds of configurable parameters. A datacenter with just a few hundreds of servers would typically have millions of various inter-dependencies. Custom application modules (e.g., SAP, Java, Job schedules; various scripts) can easily require person years to understand from scratch even for a single server. At the same time, people who create and maintain these environments change job roles, die, leave companies, and forget their knowledge over years. As a result, large portions of enterprise IT are terra incognito for IT staff.

Not surprisingly, a lot of enterprise IT components are unused but are maintained and consume electricity because nobody knows what is used and what is not. A recent study of the number of unused servers concluded that 30% of enterprise servers are comatoze on average [1]. We ourselves observed large datacenters with 50% of the servers (physical plus virtual) being unused. Nevertheless, such unused servers are being maintained and paid for: labor, software licenses (hundreds of thousands of USD per server image is not uncommon), electricity, raised floor space, data storage are just the obvious wasted resources.

Virtualization, containerization, cloudification, etc. [15] ease the pain of extra power consumption and wasted raised floor space. Unfortunately, corporate datacenters still widely rely on old physical servers and our goal is to have a solution usable right away without the costly and lengthy transformations. Moreover, a server (physical or virtual) may be used but some of its software components, storage, etc. may be unused. Gartner estimates that 80% of IT budgets are spent on keeping existing IT running [3]. Therefore, large fraction of all IT budgets worldwide is wasted on unused IT components.

Not only unused IT portions waste money to keep IT running but they also make IT transformation and optimization projects harder and pricey. Moving servers to a cloud or to a different datacenter; data loss prevention projects; defining secure perimeters; compliance with standards such as PCI DSS; identification of single points of failure; making disaster recovery plans, upgrading; investigating license compliance; etc.–all such initiatives are more expensive and lengthier because of the efforts wasted on handling unused IT assets. Moreover, unused IT increases the surface for hacker attacks.

In this paper, we investigate a general method to identify unused IT resources based on IT component topological inter-dependencies and propagation of usage information along the topological graphs. We compare our approach with the server resources utilization measurements-based approaches.

II. DESIGN

It is rarely possible to tell if an IT resource is being used or not by observing only that resource. For example, if a server has active software that consumes CPU, memory, generates I/O that server may be doing periodic batch job processing that nobody needs anymore. The opposite is also true: an idle server that consumes no resources for months may be a stand-by server for high availability or disaster recovery, a test server or user acceptance test/quality assurance/staging server, or a server, which use is planned in the future.

An IT resource is useful if there is a business user or consumer of that resource. For example, if there is a user workstation connecting to a web front-end and that front-end redirects requests to an SAP system that, in turn, uses a database server, and a job scheduling software on yet another system moves data to a backup server from the database server we can infer that all involved systems are used. Overall, IT resources are connected in a large complex graph with millions of dependencies and a single request from a user can involve chains of dozens of servers.

Even unused servers are highly connected because basic infrastructure software such as anti-virus, backup, etc. (typically dozens of installations) are installed as part of building any corporate server image. Figure 1 shows a real-life completely unused server with the servers it communicates with (server names are replaced with the server function names). As a result, every server in a data center (including unused servers) is connected with many other servers via infrastructure servers. Therefore, simply looking for unconnected servers or groups of servers is not a viable solution to identify unused servers.

Do not be misled by the simplicity of Figure 1. Topologies that involve just hundreds of servers are highly interconnected and complex. Figure 2 shows a server-to-server level connectivity graph with the infrastructure dependencies (most dependencies) shown with almost invisible transparent lines.

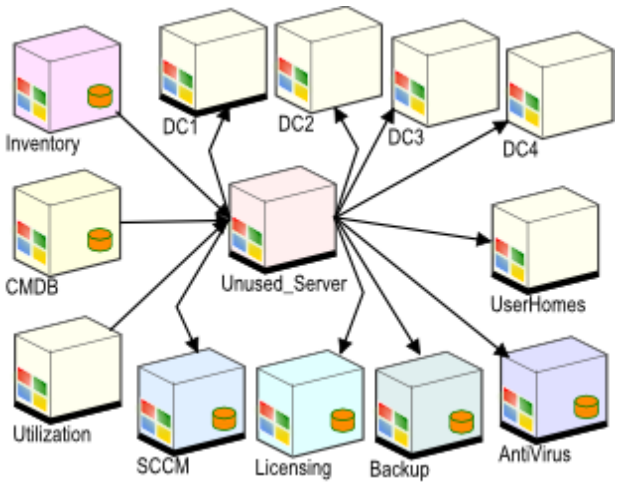

Fig 1. One example unused server in a corporate data center with the immediate servers it communicates with: Domain Controllers (DC1, DC2, DC3, DC4), a file server (UserHomes), AntiVirus and Backup servers, Licensing server, management server (SCCM), and numerous monitoring servers: Inventory, Change Management Data Base (CMDB), Utilization.

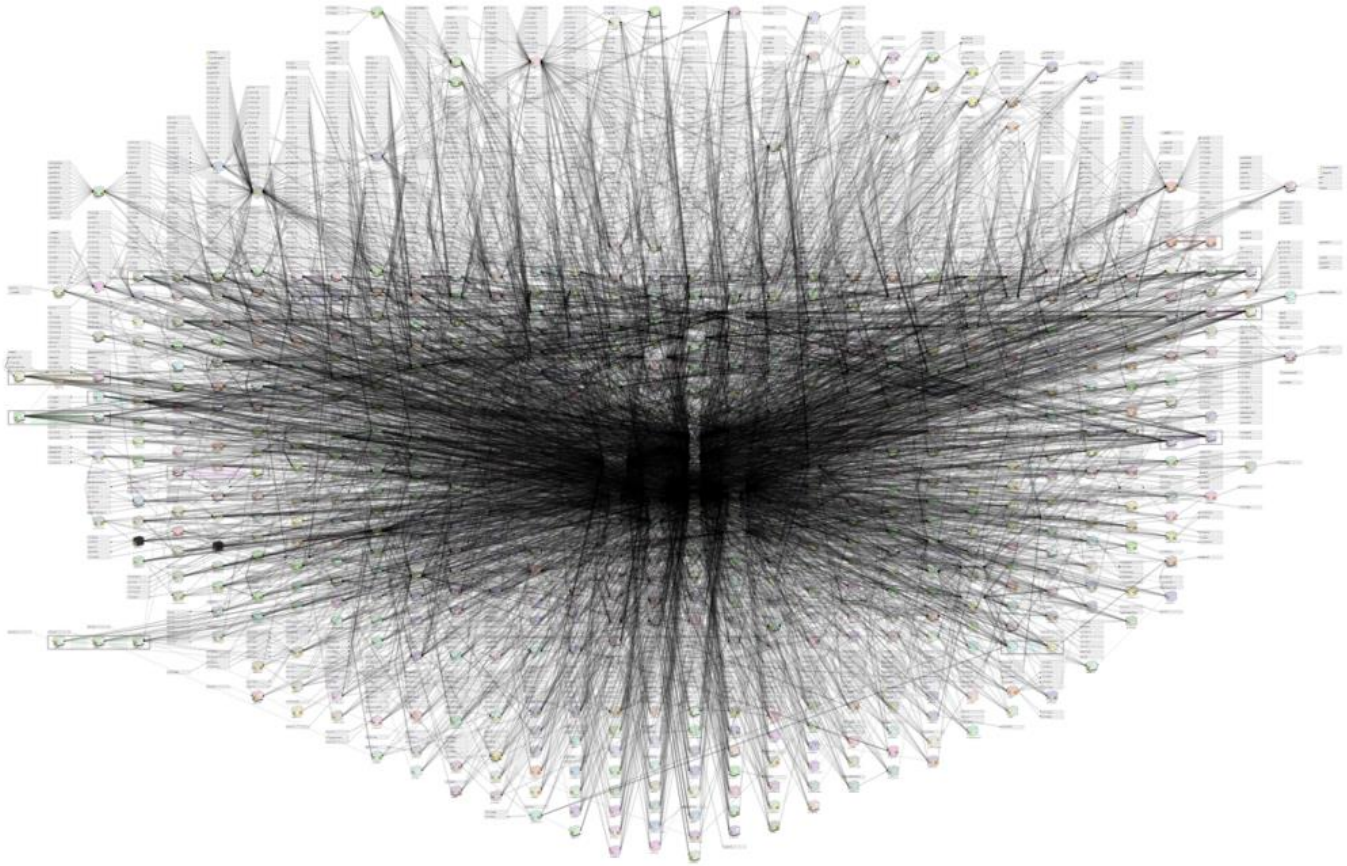

Fig. 2. A server-to-server connectivity graph for 516 servers.

A. Algorithm

Our approach to discover unused servers consists of the following main steps (illustrated in Figure 3 below):

1. Construct dependency graph including entry points from business users (e.g., workstations). Entry points can have attributes like "known to be used" or "may be used".
2. Classify graph nodes and edges (e.g., connections) as infrastructure and non-infrastructure. (This is typically accomplished via looking up a knowledge base.)
3. Starting from entry points follow non-infrastructure graph edges (including contains-type dependencies) as far as possible and follow infrastructure graph edges by one hop only (to mark infrastructure nodes too) and mark encountered nodes (e.g., as used or maybe used).
4. IT resources not marked (and resources they contain) should be considered likely unused and listed out for further verification (typically with the server owners).

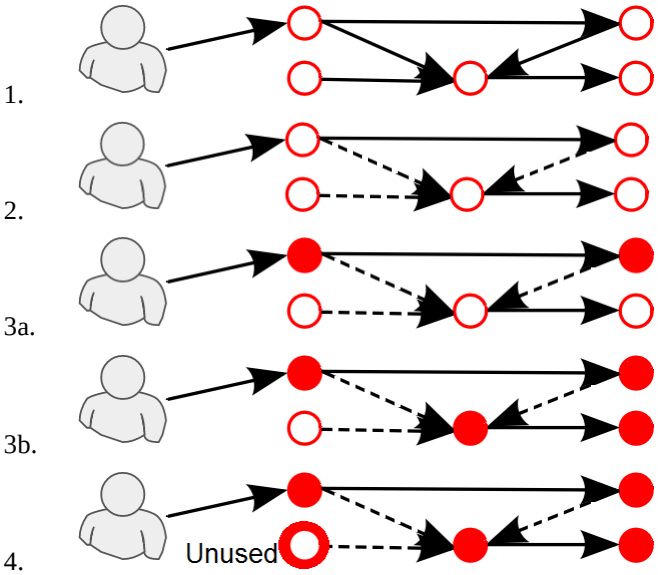

Fig 3. Steps to identify unused servers. (Infrastructure dependencies are shown dashed, nodes marked as used are shown filled.)

Traditional approach to construct server components dependency graphs relies on 1) network connections tracing and 2) analysis of software configuration files and logs. Network connections observed on the switches or on the servers are mapped to processes. Processes, in turn, are mapped to software instances. To increase the resolution it is necessary to discover internal software objects (such as databases or file shares as illustrated in Figure 4) and use software-specific mechanisms to map connections to these internal objects. We detect entry points from business users into the graph by 1) identifying connections from machines that are not known as servers and 2) analyzing connectivity logs. Unfortunately, very few servers have software connectivity logs in real life.

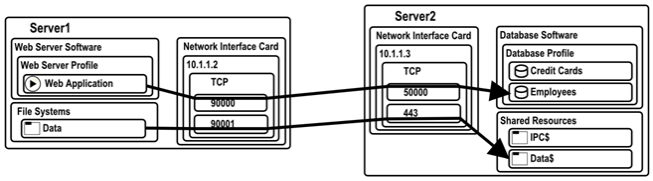

Fig. 4. Connectivity between server software components.

B. Key quality-affecting factors

Monitoring length affects 1) the graph construction quality based on the connections monitoring and 2) identification of entry points used by business users (e.g., an accountant may connect to some servers only once a quarter). 100% quality would require infinitely long monitoring. Fortunately, 95% quality of the graph construction is typically achievable within just a few weeks of monitoring with marginal improvements thereafter [4]. However, one-time sampling of connections used by most IT dependencies discovery tools is not sufficient.

Modeling of the disaster-recovery and backup systems based on their configurations is critical for the graph construction because they do not establish connections most of the time.

Resolution of internal software objects discovery critically influences the unused servers discovery. Many database and web servers are used by multiple business applications. For example, if two servers A and B connect to two different databases on the same database server C and only A is being used by a business user either all 3 are declared used (if dependencies resolution is per-server) or only A and C but not B (if dependencies resolution is per-database).

Classification of infrastructure dependencies is not as easy as it may look. A naïve classification based on the ports covers only the easiest cases (e.g., DNS). However, even recognizing which software is communicating is not enough. For example, an infrastructure component may be implemented as a web server module or be hidden (e.g., [22]). Thus, classification based on the internal middleware details is necessary.

III. IMPLEMENTATION

Our implementation is layered and consists of 1) the data collection, 2) classification, 3) graph-analytics, and 4) visualization layers. The servers-based data collection requires access to the clients' servers. Therefore, we support the most common data collection tools and CMDBs as the data collection layer in case these tools are deployed with the client. Unfortunately, existing data collection tools rarely correspond to the requirements outlined in Section II.B. Therefore, we also had to implement our own data collection layer. It monitors connections for any period of time, supports easy customization mechanisms that we use to reach 100% process and software recognition rates, and collects information with high-resolution from all common middleware systems (clusters, databases, application servers, messaging middleware, web servers, file systems and file servers, etc.) on all common operating systems. Classification of infrastructure software, objects, and dependencies is performed on top of the collected data without any interaction with the client systems. We constantly add new signatures and data collection capabilities as we use the tools.

IV. EVALUATION

We start from comparing our business-use-driven approach with the resource-utilization-based approach. We measured CPU, network and disk I/O characteristics on 516 corporate servers running Windows OS. (Servers in a datacenter of a Fortune-500 company.) Figure 5 shows Cumulative Distribution Functions of the average and maximal CPU, network and disk I/O characteristics. It is easy to see that resource utilization on most servers is low on average: 90% of the servers have below 10% CPU utilization and 30% of the servers transmitted below 1 packet/second. Low utilization complicates differentiation of used and unused servers.

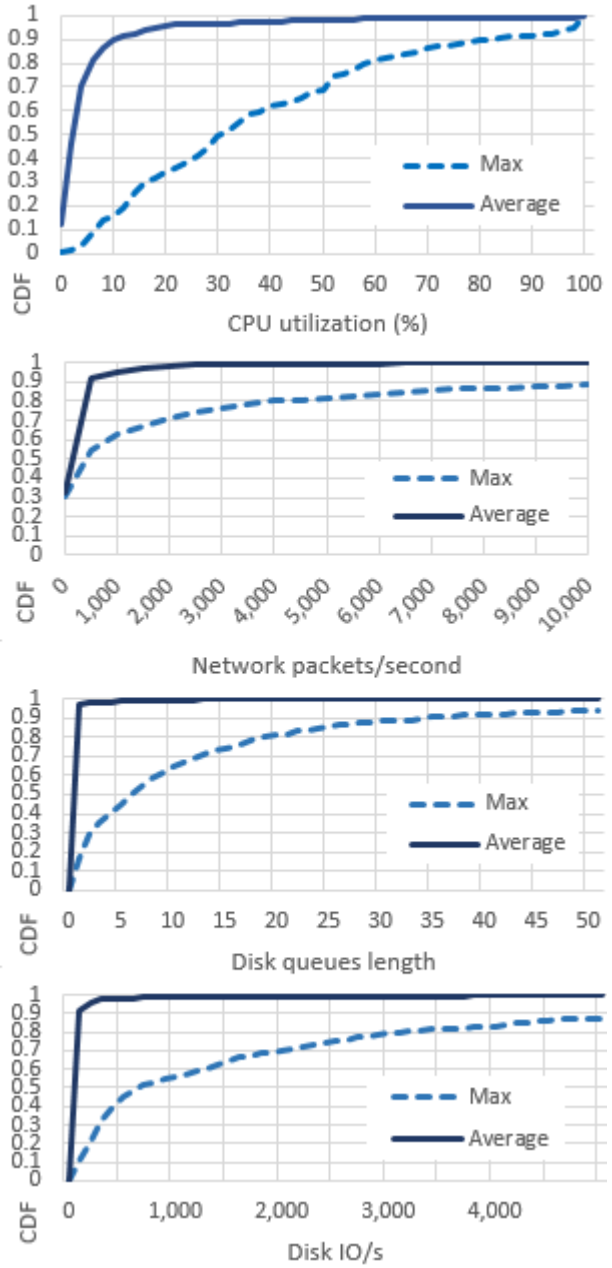

Fig. 5. Cumulative Distribution Functions (CDFs) of CPU, network and disk I/O utilizations on 516 corporate servers.

We analyzed said 516 servers using utilization and topology-based methods and discussed every server with the IT staff (application owners, server admins, etc.) to confirm every server's function. As a result, we identified 75 servers (15% of 516) that were truly unused and should be decommissioned. Figure 6 illustrates how varying threshold of CPU utilization and network packets per second to decide what servers to classify as used or not influences the amount of false positives and negatives. At any reasonably low false negatives rate (when most unused servers are detected, which is our goal) the rate of false positives gets so high that most servers are declared unused. Other I/O-related metrics produce similar results with high error rates. This may be explained by the utilization noise of infrastructure software and OS itself even on the unused servers (e.g., periodic anti-virus scans). Therefore, let us look at Figure 7 that depicts false positives and negatives rates depending on the threshold of per-software instance CPU utilization. CPU utilization of infrastructure software and OS processes was excluded from the analysis. This time even at the threshold value of zero the false negatives rate is low because a large number of servers had no useful software installed (except infrastructure). Note that the false positives rate is not zero because even on unused servers software instances perform background jobs.

Nevertheless, the rate of false positives is still high. This is because many corporate servers behave like unused servers: test, quality assurance, staging, stand-by, disaster recovery, or simply used but not utilized servers, etc.

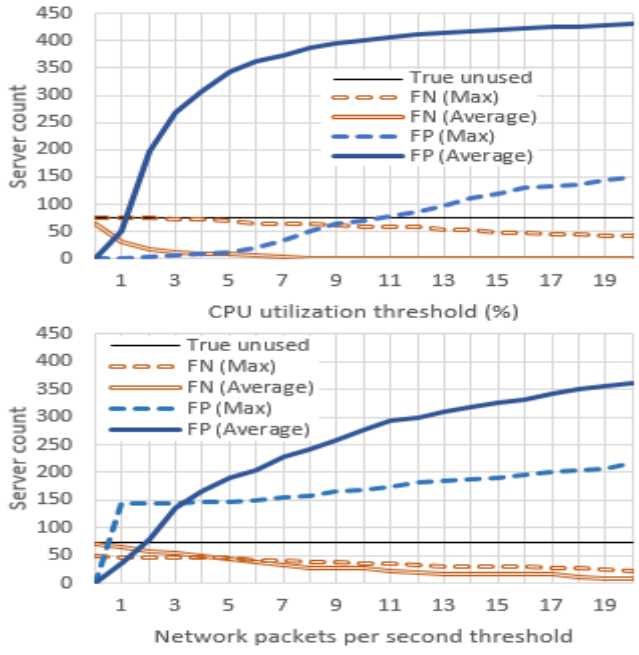

Fig. 6. False Negatives (FN) and False Positives (FP) during the detection of unused servers based on CPU (top) and network packets/second (bottom).

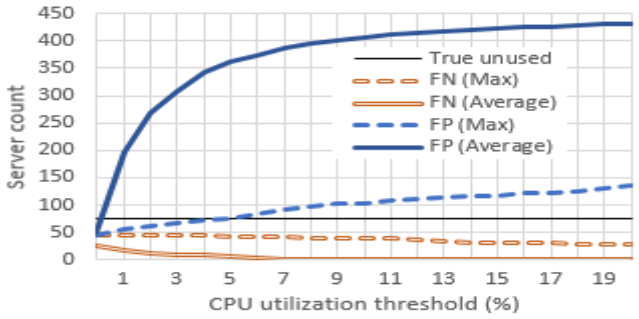

Fig 7. False Negatives (FN) and False Positives (FP) during the detection of unused servers on non-infrastructure CPU.

Topological discovery can distinguish stand-by servers in a cluster and disaster recovery servers as being used even at zero utilization. Moreover, usage logs in the middleware systems and business users' connectivity is a very reliable indicator of servers' usage. Figure 8 shows the number of false positives and negatives for topological business-use-based discovery of unused servers. Left bars show the numbers for the topological analysis per-server: If another server (not necessarily used by a business user) uses a server it is considered used. Right bars show the numbers for the topological analysis based on the business-use propagation algorithm described in Section II. Our investigation showed that 7 servers (9% of all unused servers) not detected by any topological algorithm are the false negatives due to the dependencies from servers in other datacenters. We defined business users as any node connecting from outside of the datacenter's server networks. It is possible to define any nodes in other datacenters (e.g., by subnetwork) as nodes with unknown business use and propagate this information along the graph. This way, the 7 servers would be marked as unknown to be used or unused. This deeper analysis depends on the amount of time one is ready to spend to define business users' details. 28 servers (37% of truly unused) not detected by the naïve topological one-hop algorithm are due to the groups of servers that are connected but are not used. False positives are mostly due to the test, quality assurance, pre-production, and staging servers, that were not used during the weeks when we collected the data. Multiple months of monitoring can decrease the false positives rates. However, the numbers measured are already sufficiently low for the targeted interviews with the IT staff.

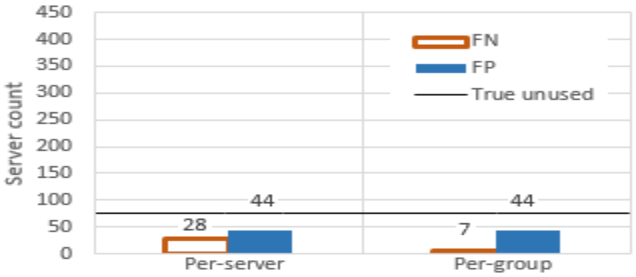

Fig. 8 False Negatives (FN) and False Positives (FP) during the topology-based detection of unused servers (per-server and per group of servers).

V. BACKGROUND

A number of off-the-shelf (e.g., [10-13]) and research (e.g., [16]) tools map network connections to software to generate IT dependency graphs and exist on the market for more than a decade. These tools and their CMDBs can be used for step 1 of our algorithm.

Most research projects focus on uncovering transaction paths by statistical correlation or instrumentation (e.g., [5-9]), which is opposite to our approach.

Business applications modeling and statistical analysis is critical for server migrations (e.g., [17, 18]).

VI. CONCLUSIONS

Datacenters consume 1-2% of all electricity in the US [2]. Delivering that electricity to the servers and other equipment reliably and removing generated heat is expensive. Moreover, modern enterprises critically depend on their IT for operation. However, 80% of corporate IT budgets is spent on maintenance of the existing IT assets [3]. At the same time, significant portion of existing servers is unused [1].

Unfortunately, enterprise IT is very complex and its objects are highly interdependent. Easy approaches that ignore interdependencies are easy but have high error rates. Graph-based approaches are intuitively better but too complex in reality. Modeling any IT portion as a graph involves modeling thousands or millions of interdependent nodes. Not surprisingly, historically the main solution path was to try to discover relevant dependencies only and analyze the graphs based on them. This is hard because the enterprise IT is too diverse and identifying and modeling sufficient number of use-defining dependencies is too hard. Our main contributions are:

1. Our algorithm is based on the detection of all types of dependencies and classification (recognition) of only the infrastructure dependencies. This is a practical and easily per-client customizable solution that is much easier to implement compared with the approach of identification of only (but all) use-related dependencies;
2. We compared the usage graph-based algorithm with the naïve utilization-based approaches. We demonstrated that utilization-based approach either does not detect too many unused servers or classifies too many used servers as unused. Therefore, without proper verification any portion of the servers can be declared unused. At the same time, our approach was shown to successfully detect more than 90% of unused servers with less than 10% of total servers misidentified as unused.

Server decommissioning is a constant and ongoing (and typically manual) process in most corporate IT environments. The proposed topology-based analysis due to its low error rates has the opportunity to reduce the process to running a tool and a straightforward interview to verify the findings.

Lastly, the method proposed in the paper can be used in almost any other IT domain on top of the existing topological graphs constructing tools (e.g., for networking devices, security zones [19], storage [20], and program code [21]).

REFERENCES

[1] J. Koomey and J. Taylor, New data supports finding that 30 percent of servers are Comatose, indicating that nearly a third of capital in enterprise data centers is wasted, June 2015, http://anthesisgroup.com/wp-content/uploads/2015/06/Case-Study_DataSupports30PercentComatoseEstimate-FINAL_06032015.pdf

[2] J. Koomey, My new study of data center electricity use in 2010, July 2011, http://www.koomey.com/post/8323374335

[3] C. Pettey, Gartner Says Eight of Ten Dollars Enterprises Spend on IT is "Dead Money", October 9, 2006, http://www.gartner.com/newsroom/id/497088

[4] N. Joukov, V. Shorokhov, and D. Tantsuyev, Security Audit of Data Flows across Enterprise Systems and Networks, *The 9th Intl. Conf. for Internet Technology and Secured Transactions (ICITST'14)*, London, UK, December 2014.

[5] M. Schmid, M. Thoss, T. Terrain, and R. Kroeger, A generic application-oriented performance instrumentation for multi-tier environments, *in Internaltional Symposium on Integrated Network Management*, 2007.

[6] B. C. Tak, C. Tang, C. Zhang, S. Govindan, B. Urhaonkar, and R. N. Chang, vPath: precise discovery of request processing paths from black-box observations of thread and network activities," *In Proc..USENIX Annual Conference*, June 2009.

[7] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharon, "Performance debugging for distributed systems of black boxes," *in Proc. ACM Symposium on Operating Systems Principles*, 2003.

[8] B. Sengupta, N. Banerjee, A. Anandkumar, and C. Bisdikian, "Non-intrusive transaction monitoring using system logs," *in IEEE Network Operations and Management Symposium*, April 2008.

[9] X. Chen, M. Zhang, Z. M. Mao, P. Bahl, Automating Network Application Dependency Discovery: Experiences, Limitations, and New Solutions, *in Proc. Of USENIX Symposium on Operating System Design and Implementation (OSDI'08)*, San Diego, 2008.

[10] IBM, Tivoli Application Dependency Discovery Manager, http://www-03.ibm.com/software/products/en/tivoliapplicationdependencydiscoverymanager/

[11] HP, Universal CMDB, http://www8.hp.com/us/en/software-solutions/software.html?compURI=1172882

[12] BMC Software, Atrium Discovery and Dependency Mapping, http://www.bmc.com/products/application-mapping/discovery-dependency-mapping.html

[13] VMware, vCenter Application Discovery Manager, http://www.vmware.com/products/application-discovery-manager

[14] Q. Wu, Making Facebook's software infrastructure more energy efficient with Autoscale, August 2014, https://code.facebook.com/posts/816473015039157/making-facebook-s-software-infrastructure-more-energy-efficient-with-autoscale/

[15] K. Rajamani, W. Felter, A. Ferreira, and J. Rubio, Spyre: A Resource Management Framework for Container-Based Clouds, USENIX Container Management Summit, November 2015.

[16] N. Joukov, M.V. Devarakonda, K. Magoutis, and N. Vogl, Built-to-Order Service Engineering for Enterprise IT Discovery, *IEEE International Conference on Services Computing (SCC'08)*, Honolulu, Hawaii, July 2008.

[17] B. Pfitzmann, and N. Joukov, Migration to Multi-Image Cloud Templates, *IEEE Intl. Conference on Services Computing (SCC 2011)*, Washington, DC, July 2011.

[18] M. Nidd, K. Bai, J. Hwang, M. Vukovic, and M. Tacci, Automated Business Application Discovery, *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, ON, May 2015.

[19] H. Ramasamy, C. Tsao, B. Pfitzmann, N. Joukov, and J.W. Murray, Towards Automated Identification of Security Zone Classification in Enterprise Networks, *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE '11)*, Boston, MA, March 2011.

[20] N. Joukov, B. Pfitzmann, H.V. Ramasamy, and M.V. Devarakonda, Application-Storage Discovery, *The 3rd Annual Haifa Experimental Systems Conference (Systor'10)*, Haifa, Israel, May 2010.

[21] N. Joukov, V. Tarasov, J. Ossher, S. Chicherin, M. Pistoia, T. Tateishi, Static Discovery and Remediation of Code-Embedded Resource Dependencies, *IFIP/IEEE International Symposium on Integrated Network Management (IM'11)*, Dublin, Ireland, May 2011.

[22] A. Traeger, N. Joukov, J. Sipek, and E. Zadok, Using Free Web Storage for Data Backup, *In Proc. of the second ACM International Workshop on Storage Security and Survivability (StorageSS 2006), in conjunction with ACM Conference on Computer and Communications Security (CCS 2006)*, pp. 73-77, Alexandria, VA, October 2006.