


E-Government Services Migration to the Public Cloud: Experiments and Technical Findings

Taavi Kotka¹, Bruce Johnson², Tomaz Cebul², Luka Lovosevic², and Innar Liiv¹

¹ Department of Informatics, Tallinn University of Technology, 12618 Tallinn, Estonia
taavi.kotka@gmail.com, innar.liiv@ttu.ee

² Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98103, USA
{bjohnson,tomaz.cebul,luka.lovosevic}@microsoft.com

Abstract. E-government services migration to the public cloud presents novel policy and technical challenges. This paper explores possible technical obstacles one should anticipate when migrating e-government services to cloud. Technical experiment design is presented, implementation process and the steps taken are elaborated; performance experiments are presented together with findings that were considered significant in the process. Main findings of migration experiments are organized into six groups: security, identity and data architecture findings; operations architecture findings; application architecture findings; compute architecture findings; storage architecture findings; and network architecture findings.

1 Introduction

Current research project sought to understand how cloud based application packaging is able to help overcome the environmental dependencies that would have previously prevented the on-premises applications from being moved to either a different physical or online location.

Public cloud platforms provide the first real steps towards abstraction of the physical computing environment, which includes servers, networking equipment, and storage systems, through the use of different cloud application packaging types. Cloud-based application packaging can take the form of specialized installers, containers, virtual disks, or entire virtual machines. Virtual machine and virtual disks can be used as a type of cloud application packaging to virtualize physical host and hardware. They may also be used to capture system specific settings including device drivers, network interfaces and Internet Protocol (IP) addresses, routing paths, DNS settings for both hosts and subdomains, cryptographic keys, user and machine credentials as well as many other application specific settings. In the context of this project it was also proposed that the use of cloud application packaging would enable consolidation and portability of applications, as well as make the operations, maintenance, and application development simpler tasks by mitigating some of the more disruptive elements in application lifecycle management.

Furthermore, it was assumed a major benefit of a Virtual Data Embassy Solution [1] would be a consistent (seamless) online environment based on the latest versions of compatible hardware and software. The Solution is expected to eliminate the variability

of stand-alone hardware and software environments, which have been developed (and upgraded) at different intervals over time. For example, when each application is free to dictate all layers of the software and hardware stack, there is a real risk that it can be extremely complex to move that application to a new physical, e.g. physical embassy, or cloud environment, e.g. Virtual Data Embassy. The research project looked at how, when using a public cloud platform, significant layers of the application stack could be consolidated, standardized, and scaled in ways that make it possible to move and operate all applications in a more standardized and repeatable fashion.

Finally, it was important to show how the use of a public cloud platform could help optimize for different sets of skill and operational models required by administrators of e-government services, building a critical mass of knowledge overall. To this end, the research project team sought to show how a basic IaaS platform could support two different applications using common operations up to the level of the specific application requirements. While it was assumed that the use of an IaaS platform would not completely eliminate the need for subject matter experts, it was found to reduce the amount of support needed. Moreover, software automation allows resources, previously allocated to support on premise applications, to be redeployed in other areas.

Following sections explore how the Virtual Data Embassy could be implemented within the context of the current Estonian government ICT architecture [1] (see [2] for discussion about policy and legal aspects). The implementation process and the steps taken are elaborated, followed by a section that compares and contrasts the results of the testing that was conducted, before ending with findings that were considered significant in the process.

2 Technical Experiment Design

At the onset of the research project a number of steps needed to be taken, from which a number of implications for the project as a whole, were derived. While not part of this document, the process of selecting from the different cloud options available and selecting the government services to be migrated was critical to the success of the research. To this end comprehensive risk assessments were conducted, which allowed the selection of the most appropriate services, as well as highlighting a number of opportunities for improvement overall.

Building on this, consideration was given to how to best migrate the services to the cloud. The challenges encountered in this process, as well as solutions used, are presented in the section below. The next section talks about how the different starting architectures across the two project workstreams were worked around in order to ensure that in the cloud operational efficiencies were achieved, e.g. common file storage, common backup procedures, and common load balancing technologies. Lastly, an overview of the testing conducted, once migration had been completed, is presented. Comparison of different cloud platforms was not in the focus of this project.

2.1 Migration to the Public Cloud

The migration of existing e-government services to a public cloud platform, and the feasibility of doing so has recently gained substantial interest and attention [3–5]. Related practical experiments represented a major part of the research project. If it had emerged that the migration process was too complex, costly, time consuming or that it required significant architectural changes, doubts would have been cast on the overall viability of the Virtual Data Embassy Solution. An assessment was therefore made at the beginning of the project to understand it would be feasible and whether any significant changes would be required.

Two main approaches were considered for migrating the selected government services, although it has to be pointed out that these cannot be seen as binary either/or options. It is expected that in other similar situations a combination of these two would be used, as one is a better fit for newer operating system migration and the other for more complex applications:

- **Virtualize the application and perform an “in-place” base operating system upgrade:** This approach consists of three distinct steps: (1) any physical servers are virtualized and any existing virtual servers cloned; (2) the operating system is upgraded to the latest version in-place; and, (3) all is uploaded onto the cloud platform. Typically, this approach is beneficial if there is a need to maintain a direct clone of the entire operating system environment on-premises. However, this approach requires significantly more time and bandwidth since the images to be uploaded can be large, e.g. over 50 GB.
- **Deploy directly onto the cloud platform:** This approach sees the base operating system provisioned directly by the cloud platform before any application components, such as databases or web servers, are deployed onto the operating system and the application content and data is synchronized. This approach has the benefit of requiring only the core application and its data to be uploaded to the cloud which frequently represents a much smaller data footprint.

For the migration to be possible, it had to be established whether the Microsoft cloud platform supported the operating systems currently used by the government services and their applications. As highlighted in the research project description, the two government services use FreeBSD and CentOS, which are not Microsoft products. Importantly, the cloud platform selected supports a number of non-Microsoft products, including UNIX & Linux operating systems, as well as FreeBSD and CentOS. However, it typically only supports the latest three versions of an operating system. This means that an operating system upgrade was required prior to migration, as the versions used by the selected applications were not supported. Furthermore, on-premises virtual machines can also be transferred onto the Microsoft’s cloud platform directly, as it supports the open industry-standard virtual hard disk (VHD) format [6]. This is used by a number of on-premises hypervisors.

Given the base operating system upgrades required by both application workstreams, the second option, “deploy directly in the cloud”, was selected as being quicker and less risky. One reason for this choice was that the software installation media for the

application components was readily available, making it straightforward and fast to re-install them on the public cloud operating system. The second reason was that this option meant the content and data could be synchronized directly with the on-premises master, again with speed and ease. Finally, and most critically, the versions of FreeBSD and CentOS used originally did not natively support an in-place operating systems upgrade. As a result, the in-place upgrade, core to the first option, would have to be performed manually, introducing a higher level of risk.

2.2 Target Architecture for Public Cloud

While the two project workstreams had different starting architectures, the research project aimed to deploy the applications in a similar deployment architecture to help achieve operational efficiencies, such as common file storage, backup procedures, and load balancing technologies. Target deployment architectures were utilized as presented in following subsections.

2.2.1 Workstream #1 – Presidentee.cloudapp.net

Presidentee.cloudapp.net was a Microsoft cloud application version of www.presidentee.com site. The cloud service “presidentee.cloudapp.net” represented six virtual machine instances behind a load balancer configured in an availability set. The cloud service had been configured to use auto-scaling with minimum of two instances to a maximum of six instances. It is important to point out that two instances represent a minimum required to be able to achieve the level of availability needed. The overall design is depicted in Fig. 1 below.

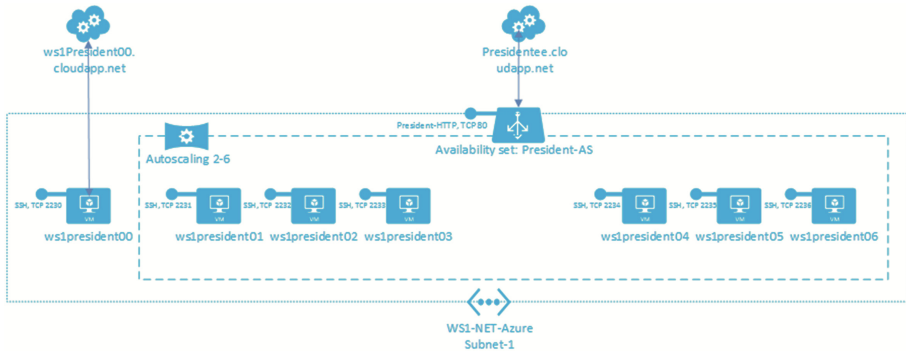


Fig. 1. Presidentee.cloudapp.net architecture

The cloud service had HTTP protocol on TCP port 80 published on the internet. The load balancer ensures that the network traffic is equally divided between all running virtual machine instances. Moreover, when a new virtual machine instance starts or

stops, it is automatically included in the load balancer pool. As mentioned above, the cloud service was serviced by up to six virtual machines kept on geographically replicated storage and configured as an availability set.

All virtual machines used were sized Standard A1 (1 core, 1.75 GB memory), had a fixed IP reservation and a single disk sized 50 GB. The IP address was reserved in the WS1-NET-Azure Subnet 1 IP address segment. As a result, the cloud platform fabric always assigned the same IP address to the virtual machine. Finally, the content was synchronized from on-premises publishing servers using the custom rsync protocol over the Secure Shell (SSH) to ws1president00 server and, in the second stage, from ws1president00 to ws1president(01-06).

2.2.2 Workstream #1 – Riigiteataja.cloudapp.net

Riigiteataja.cloudapp.net was the portable version of www.riigiteataja.ee site running on the Microsoft cloud platform. The cloud service “riigiteataja.cloudapp.net” represented four virtual machine instances behind a load balancer configured in an availability set. The cloud service had been configured to use auto-scaling with minimum of two instances to a maximum of four instances. It is important to point out that two instances represent a minimum required to be able to achieve the level of availability needed. The overall design is depicted in Fig. 2 below:

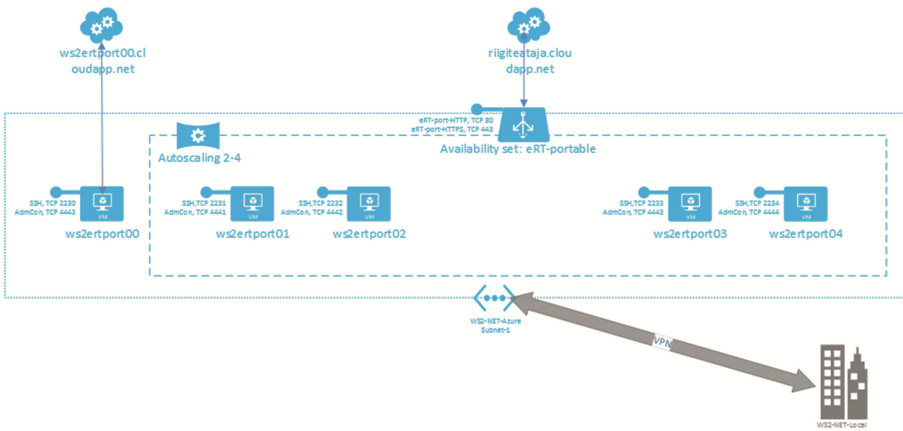


Fig. 2. Riigiteataja.cloudapp.net architecture

The cloud service had both HTTP and secure HTTPS protocol on TCP port 80 and 443 published on the Internet. The load balancer ensures that network traffic is equally divided between all running virtual machine instances. When a new virtual machine instance starts or stops, it is automatically included in the load balancer pool. As

mentioned above, the cloud service was serviced by up to four virtual machines stored on geographically replicated storage and configured as an availability set.

All virtual machines used were sized Standard A5 (2 cores, 14 GB memory), had a fixed IP reservation and two disk drives (OS disk sized 30 GB, data disk sized 200 GB). The IP address was reserved in the WS2-NET-Azure Subnet 1 IP address segment. As a result, the cloud platform fabric always assigned the same IP address to the virtual machine. For synchronization purposes, a site to site Virtual Private Network (VPN) was also created between the WS2-NET-Azure virtual network and the network of the Ministry of Justice. Moreover, the cloud service `riigiteataja.cloudapp.net` was also assigned a reserved public IP address to make external DNS handling easier. Finally, the content was synchronized from on-premises publishing servers using custom `rsync` protocol over SSH to `ws2ertport00` server and in the second stage from `ws2ertport00` to `ws2ertport(01-04)`.

2.3 Performance Testing Methodology

Once the migration of the two applications was completed, testing activities began across two dimensions: performance and demand. Two types of performance tests were used: load and stress testing. Load testing was used to understand the behavior of the system under a range of normal load conditions, and to compare the transactional response time with the on-premises solution. Stress testing, on the other hand, was used to determine the solution's robustness under peak load and to prove it would automatically scale-out elastically under sustained peak load situations.

Load testing involves simulated client and end user activity that could take place, if a large number of human end users were attempting to access the services at one time. The patterns and usage scenarios are designed to ensure that the correct activities are available to users. For example, if one million users were to go to the president's website due to an external event, this would be considered a load test scenario. The users are not trying to do unusual but their sheer numbers could impact performance. Conversely, the stress testing scenarios tend to simulate client and end user behavior designed to break or cause problems for the site, e.g. where multiple users try and overload it by playing videos to attempt to consume all the resources available, thus preventing the site from functioning. The load and stress testing were also repeated under two demand scenarios. The first was a normal demand scenario, which consisted of replicating the existing normal usage conditions whilst catering to organic expanded demand, as might occur if additional users were to utilize the e-government service. The second was a malicious demand scenario, which might occur if Estonian e-government services were under a cyber-attack intended to render the services incapacitated or unavailable. Section 3 covers the results of the testing.

2.3.1 Normal Demand Usage Scenario

The normal demand usage scenario was to demonstrate the cloud platform's ability to dynamically scale up the number of application servers, storage systems, and route traffic appropriate to the end user demand. For example, if a text based and media content

update to the President.ee website were to occur, load testing should show that the cloud platform has supported increased demand for the media files and web server content. In an ideal situation, the cloud platform would dynamically scale up compute, storage, and network resources using the cloud platform load balancer to deliver the content to users via the closest Virtual Data Embassy, irrespective of where the data center is based.

Under normal load testing, it was expected that the cloud platform could demonstrate automatic scaling, eliminating the need for procurement, setup, or redeployment of applications by the administrator. The normal load testing scenario was also designed to show the reliability of the cloud platform under normal failure events, such as a (non-malicious) application crash due to a software bug in either the application or underlying guest operating system. If such an instance were to occur, the cloud platform should automatically use a replacement application instance with no staff involvement.

2.3.2 Malicious Demand Usage Scenario

The malicious load and stress testing was to demonstrate that the cloud platform is resilient to malicious attempts to consume compute, storage, or network resources, which would prevent a normal end user from accessing the application or monument websites within a reasonable response time. Under malicious load and stress testing, the cloud platform was expected to implement parameterized automatic scaling, which would eliminate the need for operator involvement or the procurement, setup, or redeployment of applications by the administrator.

The simulated malicious load testing scenario was designed to demonstrate the reliability of the cloud platform against malicious attacks that would attempt to exploit known (e.g. Heartbleed SSL) or unknown (e.g. zero day attacks) software bugs in either the application or underlying guest operating system. In a malicious failure scenario under load, the cloud platform should automatically use a replacement application instance with no operations staff involvement and begin to report malicious usage to key staff people for examination of possible mitigation techniques beyond simple load balancing.

3 Experiments and Performance

In this section, the performance results are outlined, comparing the applications' behavior in their original on-premises environments and the target cloud environment. All of the performance, load and stress test cases were built using Microsoft Visual Studio® 2013 and Visual Studio Online and run on the Azure™ cloud platform. The results were exported and analyzed by the testing team. Microsoft Azure™ was used to initialize a large number of distributed test agents that can successfully simulate a website load under different circumstances, such as different bandwidth, browser types, click pattern, etc. Figure 3 depicts the load testing approach and architecture.

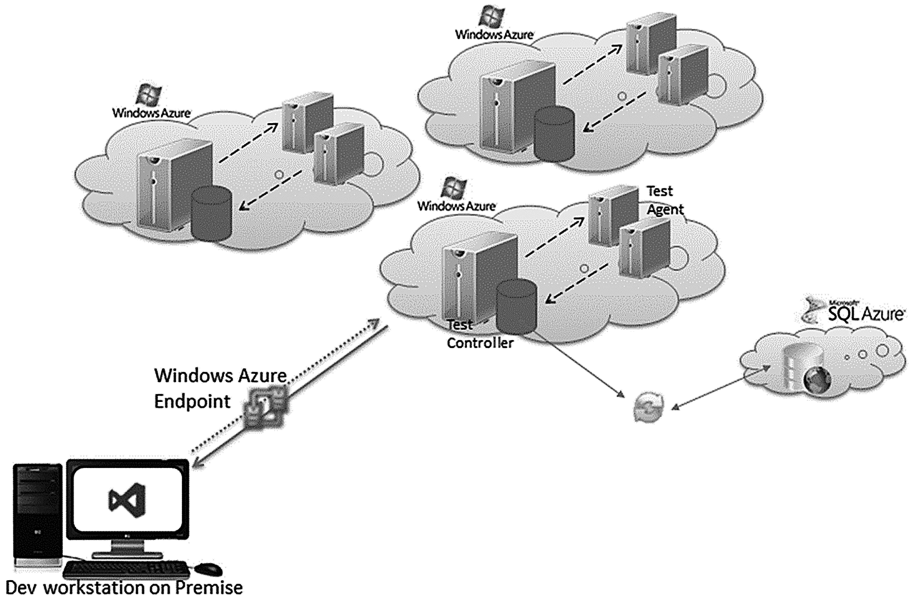


Fig. 3. Load testing architecture

3.1 Workstation #1 Load Test Results

The load tests were performed on both the on-premises and cloud versions of President.ee. Overall, the website performed as expected and the tests showed that the response time was in the proposed limit (goal) of 5 s. The tests we conducted that are particularly worth highlighting were:

- *#1.1 HomePageLoadTest*: Auto-scaling test, verifying that website scales up automatically under heavy load;
- *#1.2 SearchResultsLoadTest*: Basic performance test measuring load time of homepage and search results from the website;
- *#1.3 MediaImageLoadTest*: Media performance test measuring load time of random image and random video from the website;
- *#1.4 TestMixLoadTest*: Mixed performance test (also known as a mixed feature test), simulating a typical scenario in which the website is under heavy load in different areas: e.g. 70 % of the users on the home page, 16 % on the search page and 12 % on image content and 2 % on video content.

Table 1 represents average response times for different load test scenarios. Response times are listed for both cloud and on-premises version of President.ee.

Table 1. Overview of key tests conducted in Workstream #1

Load test	# of users	Duration of test	Azure avg. response time	On-premise avg. response time	Comments
#1.1	1000 users	30 min	3.14 s	3.38 s	Auto-scaling enabled, starting with 1 virtual machine instance
#1.2	1000 users	30 min	2.90 s	2.93 s	
#1.3	1000 users	30 min	0.20 s	<i>not tested</i>	
#1.4	500 users	30 min	3.19 s	4.08 s	Mixed tests will all features tested

Azure Workstream #1 Load Test Results: Figure 4 shows typical load test results, when run from Visual Studio® and deployed to Visual Studio Online. This is a sample load test for the image content (Load test #1.3) for 500 concurrent users during a period of 30 min. The test gave us the average page response time of around 0.2 s. It is important to note that the first few spikes in Fig. 4 were managed by the auto-scaling feature. This feature was demonstrated during the tests under heavy load by starting an additional virtual machine instance to offload the traffic (Fig. 5).

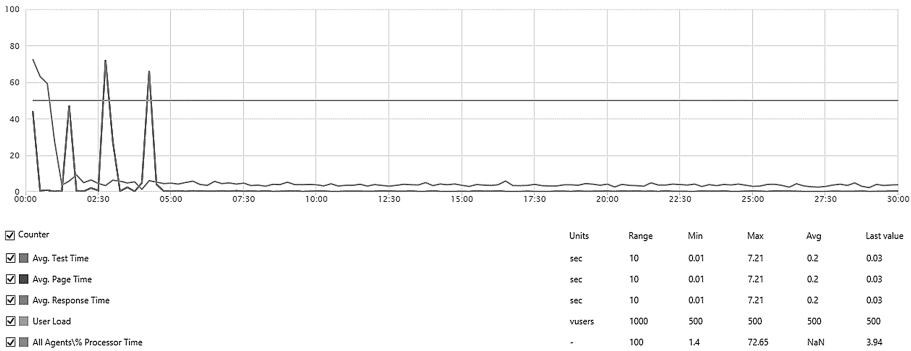


Fig. 4. Azure™ version of MediaImageLoad (#1.3) test results

On-Premises Workstream #1 Load Test Results: The on-premises version was running on the following hardware: CPU: Intel Xeon X3350 and RAM: 8 GB. The following diagram shows typical load test results when run from Visual Studio® (and deployed to Visual Studio Online). This is a sample load test for the image content (HomePageLoadTest, #1.1) for 500 concurrent users during 30 min. The first couple of spikes (Fig. 6) for the average page response time could be explained by “cold boot”; when the server/database is warmed-up, the page response time gets normalized.

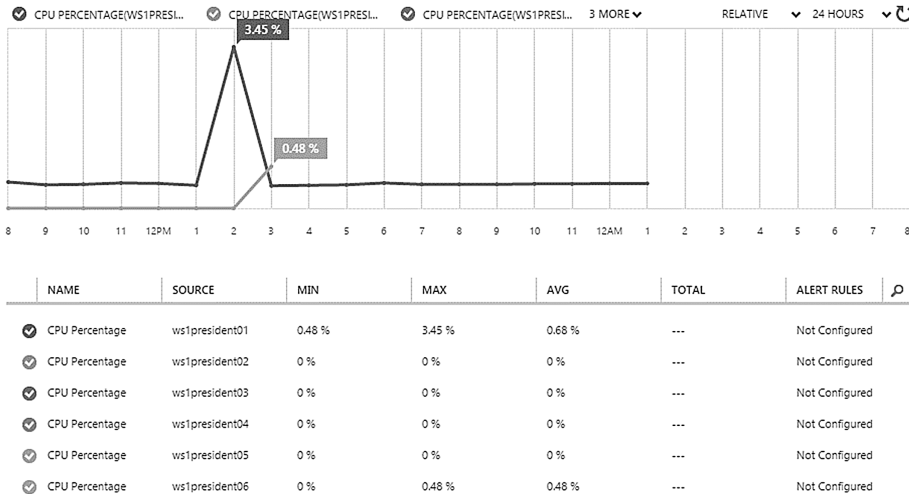


Fig. 5. Azure™ Monitoring Dashboard demonstrates auto-scaling features

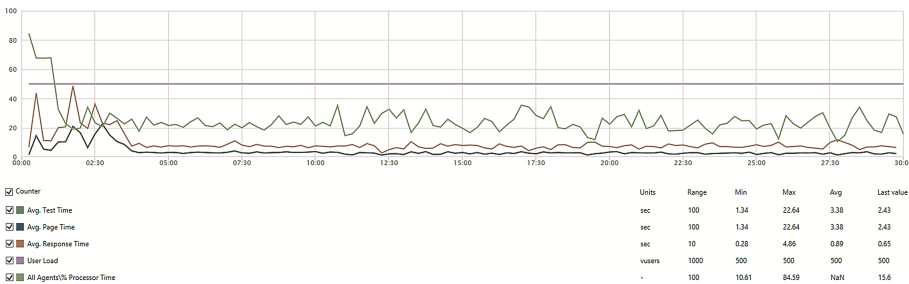


Fig. 6. Load test results on premises, Workstream #1

3.2 Workstation #2 Load Test Results

Tests were performed against a full version of the website, which had read, write, and modify functionalities enabled. The website ran both in the cloud and on-premises. As with the first workstream, the website performed as expected and the tests showed that the response time was in the proposed limit (goal) of 5 s. The tests conducted that are particularly worth highlighting were:

- #2.1 *HomePageLoadTest*: Basic performance test, measuring load time of home-page;
- #2.2 *SearchResultsLoadTest*: Measuring load time of search results from the website
- #2.3 *LawDetailsLoadTest*: Measuring load time of law/act details page

- #2.4 *TestMixLoadTest*: Mixed performance test, simulating a typical scenario in which the website is under heavy load in different areas, e.g. 50 % of the users on the home page, 30 % on the search page and 20 % on law/act details page (Table 2).

Table 2. Overview of key tests conducted in Workstream #2

Load test	# of users	Duration of test	Azure avg. response time	On-premise avg. response time
#2.1	1000 users	30 min	3.21 s	3.01 s
#2.2	1000 users	30 min	156.74 s	90.64 s
#2.3	1000 users	30 min	1.25 s	3.91 s
#2.4	500 users	30 min	72.49 s	25.28 s

Azure Workstream #2 Full Version Load Test Results: The following diagram shows a sample of load test results when run from Visual Studio (and deployed to Visual Studio Online). This is the load test for the search results page (part of the basic performance test for workstream #2, #2.2) for 500 concurrent users during 30 min (Fig. 7).

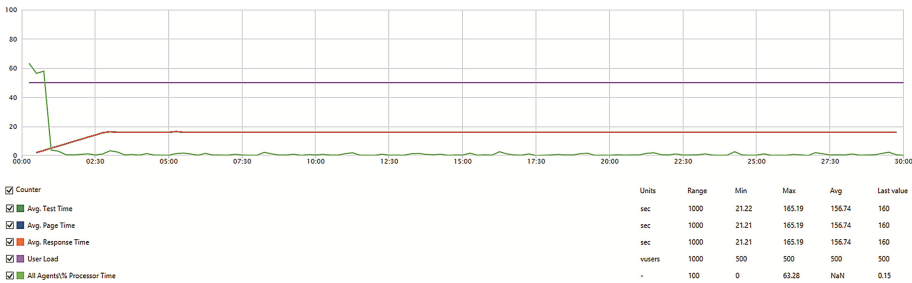


Fig. 7. Sample load test results, Azure™ workstream #2

It is worth noting that during the performance/load testing, the auto-scaling feature for the full version of workstream #2 government services on the cloud platform was not enabled. The following virtual machine sizes formed the basis of the solution: Application server (1 A5 virtual machine instance), Database server (1 A5 virtual machine instance).

On-Premises Workstream #2 Full Version Load Test Results: The following diagram (Fig. 8) shows a sample of load test results when run from Visual Studio (and deployed to Visual Studio Online). This is the load test for the search results page (part of the basic performance test for workstream #2, #2.2) for 500 concurrent users during 30 min. The following hardware was used for the on-premises solution: the application server: 4vCPU, 12 GB RAM and the database server: 4vCPU, 8 GB RAM.

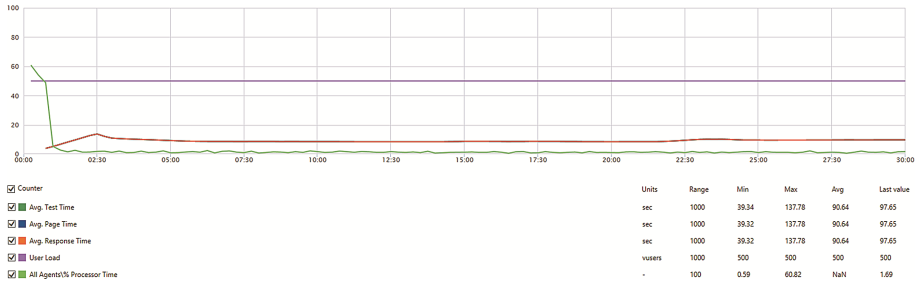


Fig. 8. Sample load test results, on-premises, workstream #2

4 Findings

Why is this paper a unique contribution to e-Government scientific community? Is this (not) yet another service migration to cloud? What are those unique aspects to e-Government services? Estonia must be able to continue to function as a government, and as a people, even in the direst of scenarios, including the loss of its territory. Since Estonia does not have paper backups of core registries, its demands for data protection, integrity, security, and privacy are unparalleled. Both experiments were conducted considering those worst scenarios, observations and findings were identified. This section presents main findings of migration experiments. They are organized into six main groups, where each group and finding would deserve a longer explanation and further discussion. However, authors see this as a guiding framework to spark academic discussion and comparisons between different migration projects. Technical reports, which describe each finding on a technical level in more detail are available to all researchers upon request.

4.1 Security, Identity and Data Architecture Findings

1. Data architecture and data security policies are essential for data integrity
2. A holistic data governance and security approach is required to facilitate migration to the cloud
3. Designing systems for ‘separation of duties’ and ‘least privilege’ is vital to maintaining security
4. Role Based Access Control (RBAC) and claims-based security increases overall security
5. Overall system control increases if operating system root credentials are restricted
6. Isolation of user roles and accounts between environments prevents accidental changes
7. Isolating service accounts between application instances reduces security risk.

4.2 Operations Architecture Findings

1. Documented disaster recovery and cloud fail-over procedures ease pressure on teams

2. Expected load characteristics and required capacity plans needed to be able to scale the cloud
3. Standardization across e-government services can ease operational management
4. Operation support roles benefit from use of Role Based Access Control.

4.3 Application Architecture Findings

1. A DevOps approach [7] to application building ensures a quicker response to threats
2. Understanding security threats for all applications is central to successful mitigation
3. A modern design, which allows for portability, gives the government more choice
4. Public cloud can protect against DDoS attacks better than on-premises systems.

4.4 Compute Architecture Findings

1. Standardized sizing of physical servers and virtual machines helps migration
2. Master operating system images can drive standardization through virtual machine templates
3. Complexity reduced if distinct functions are separated onto different compute nodes
4. Isolated content roles minimize risk
5. Cloud design patterns using small scale-out units are preferable
6. Auto-scaling configuration requires further development
7. A server patch management strategy can help improve security.

4.5 Storage Architecture Findings

1. Standard file system disk structure for database, log, and application data improves performance
2. Any modification of the overall host, disk, and file system layout is dependent on boundaries
3. The primary operating system disks cannot be used for application or service storage needs
4. Data files can be shared between application servers
5. Data replication between on-premises and cloud systems is straightforward
6. Cloud storage architecture can enhance existing application storage
7. Regular updates of cloud virtual machine operating systems needed for wider use of Windows Azure Storage functionality.

4.6 Network Architecture Findings

1. Statically configured internal DNS naming conventions are required in the cloud environment
2. Ownership, configuration and maintenance of the DNS Start of Authority (SOA) is vital
3. Statically configured IP public and private addresses need to be carefully handled

4. Virtual network, IP address and name resolution function differently in the cloud
5. Applications need to be able to use the load balancer provided by the cloud platform.

5 Conclusions

The research project confirmed that both the Estonian President's website and the State Gazette website were able to successfully migrate to and operate in the public cloud for the duration of the project. Moreover, while certain issues were encountered, it also became clear that cloud computing can be leveraged to enhance the performance and resilience of the government services, given cloud capabilities, such as DDoS protection and auto-scaling. Indeed, virtual machines in the cloud environment can be hosted and stored in numerous locations, which may be situated in different countries, or even continents. This is necessary for Estonia's digital continuity [1] requirement – the functioning of the state in any situation or emergency.

It also emerged that while most applications, originally designed for on-premises use, can be moved to the cloud “as is”, this might result in difficulties with scaling and achieving full functionality. For e-government applications to truly benefit from a migration to a cloud platform, they should be thoroughly evaluated, e.g. undergo a risk assessment, to ensure that the applications meet the current threat landscape and have a defined switching procedure and procedures for running services in cloud in place.

A critical challenge identified was that many existing information systems are poorly documented. Detailed documentation on information system architecture and functionality specifications, interfacing, or customizations is often missing and frequently only a small number of experts understand the workings of the system. This could lead to gaps in digital continuity, in particular when quick reactions are required and these experts are not available.

A very specific lesson that emerged was the need for the government to be able to request changes regarding the operation of DNS zones and records, which is critical to usage of the DNS by end users. The DNS system and certificate authority system will play a very significant role in establishing citizens' trust and their ability to get to the appropriate services during a crisis or cyber-attack. Presently, this is currently not the case, and the DNS and critical name resolution systems rely heavily on manual updates by a collection of civil servants and private companies.

Overall, the project demonstrated that from a technology perspective, the Virtual Data Embassy Solution is feasible, as only limited architectural changes would be required.

Acknowledgments. This paper is based on a section of an unpublished manuscript of a research project on “Public Cloud Usage for Government” and unpublished materials of doctoral dissertation of Taavi Kotka, authors did not receive payment for this publication. Authors wish to thank Mikk Lellsaar, Raivo Oravas, Rome Mitt, Ivo Vellend, Taavi Meos, Ardo Birk.

References

1. Kotka, T., Liiv, I.: Concept of Estonian Government cloud and data embassies. In: Kö, A., Francesconi, E. (eds.) EGOVIS 2015. LNCS, vol. 9265, pp. 149–162. Springer, Heidelberg (2015)
2. Kotka, T., Kask, L., Raudsepp, K., Storch, T., Radloff, R., Liiv, I.: Policy and legal environment analysis for e-Government services migration to the public cloud. In: Proceedings of the 9th International Conference ICEGOV 2016, Montevideo, Uruguay, 1–3 March 2016, pp. 103–108. ACM Press (2016)
3. Cellary, W., Strykowski, S.: e-Government based on cloud computing and service-oriented architecture. In: Proceedings of the 3rd International Conference ICEGOV 2009, Bogota, Colombia, 10–13 November 2009, pp. 5–10. ACM Press (2009)
4. Pokharel, M., Park, J.: Cloud computing: future solution for e-Governance. In: Proceedings of the 3rd International Conference ICEGOV 2009, Bogota, Colombia, 10–13 November 2009, pp. 409–410. ACM Press (2009)
5. Gongolidis, E., Kalloniatis, C., Kavakli, E.: Requirements identification for migrating eGovernment applications to the cloud. In: Linawati, Mahendra, M.S., Neuhold, E.J., Tjoa, A.M., You, I. (eds.) ICT-EurAsia 2014. LNCS, vol. 8407, pp. 150–158. Springer, Heidelberg (2014)
6. Microsoft: About VHD. [https://msdn.microsoft.com/en-us/library/windows/desktop/dd323654\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd323654(v=vs.85).aspx)
7. Hüttermann, M.: DevOps for Developers: Integrate Development and Operations, the Agile Way. Apress, New York (2012)