Cloud as a Platform for W-Machine Didactic Computer Simulator

Piotr Paczuła[1], Robert Tutajewicz[1], Robert Brzeski[1], Hafedh Zghidi[1], Alina Momot[1], Ewa Płuciennik[1], Adam Duszeńko[1], Stanisław Kozielski[1], and Dariusz Mrozek[1,2]

[1] Department of Applied Informatics, Silesian University of Technology
ul. Akademicka 16, 44-100 Gliwice, Poland
[2] Institute of Biomedical Informatics, National Yang Ming Chiao Tung University, Taipei City, Taiwan (R.O.C.)
dariusz.mrozek@polsl.pl

Abstract. Effective teaching of how computers work is essential for future computer engineers and requires fairly simple computer simulators used in regular students' education. This article shows the evaluation of alternative architectures (platform-based and serverless) for cloud computing as a working platform for a simple didactic computer simulator called W-Machine. The model of this didactic computer is presented at the microarchitecture level, emphasizing the design of the control unit of the computer. The W-Machine computer simulator allows students to create both new instructions and whole assembly language programs.

Keywords: teaching, computational science, computer model, cloud computing, serverless computing, platform as a service (PaaS)

1 Introduction

The complexity of computer organization and construction causes that they are usually presented at the model level. The type of model depends on the purpose of the presentation. A commonly used model is the instruction set level description, also known as the Instruction Set Architecture [12,14]. It defines the architecture of the computer and includes a description of registers, a description of memory, and a description of the execution of all instructions performed by the computer. Such a model is sufficient for compiler developers, as well as for assembly language programmers. A more detailed description of the computer organization at the level of the so-called microarchitecture takes into account the registers and functional units of the processor (the central processing unit of the computer) and the main connections between them. The microarchitecture model of the computer allows defining the instruction cycle of the processor, i.e., the successive steps of instruction execution. To present the development of computer organization, the literature proposes didactic computer models exposing selected details of the concepts being explained and simplifying other elements of computer design. Many basic textbooks, e.g. [5,12,14], use such models. While

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08760-8_56

the initial model is usually very simplified, subsequent versions of the model are extended to enable the presentation of increasingly complex real-world processor problems. For example, models Mic-1 and Mic-2 are used in [14] to explain microprogramming control of the processor, while Mic-3 and Mic-4 are used to present the pipelined execution of the instruction cycle. The model using the MIPS processor architecture, presented in the book [5] and its earlier editions, is close to the actual processor organization. This model provides a simple and very clear way to explain the concept of pipelined processor organization, including the transition from non-pipelined to the pipelined organization. It also allows illustrating the problems associated with the pipelined organization and how to solve them. These advantages have determined that the MIPS model is used as a teaching model in computer architecture lectures at many universities around the world.

The presented W-Machine, created by S. Wegrzyn and next developed by S. Kozielski as a hardware device, is a didactic model of a simple, fully-functional computer described at the instruction set level and the microarchitecture level [16]. The specific feature of the microarchitecture model is that it distinguishes all signals controlling the transmission of data and addresses during the execution of the instruction cycle of this computer. This model shows in detail the design of the control unit that generates the mentioned signals that control the buses and functional units. The model includes two variants of the control unit structure: hardwired and microprogrammed. The open structure of the W-Machine simulator allows the users to independently define new machine instructions by creating microprograms that perform the functions of these instructions. Simultaneous visualization of active processor elements during the instruction execution and the possibility to use priorly implemented instructions in the creation of own assembly language program allows and simplifies the student's understanding and learning of the idea of computer operation in the commonly used von Neumann architecture [4]. This brings to the curriculum structure elements that connect the hardware and software. In this way, the course becomes more creative and effective for both the teacher and the student by using the W-Machine simulator, which can be a very good tool to aid in teaching and learning. However, teaching the computer organization and operation on such a low level now faces many challenges. First of all, the pandemic situation changed the conditions in which we live [2], including teaching opportunities [11]. Remote work is not only becoming more and more popular but is even often required [1]. Restrictions in movement mean that the work is more often done from home. Additionally, taking into account the internationalization of universities and frequent difficulties in crossing borders by foreign students, the possibility of remote access to educational tools becomes even obligatory. Cloud computing, due to the shared resources, the range of remote access, and global availability of the platform and services, can be one of the very good solutions to deploy and distribute the educational software [3,7,15,17]. In this paper, we test two alternative approaches for disclosing the W-Machine simulator in the Cloud environment - the Platform-based (i.e., PaaS) and serverless-based.

2 Background

The W-Machine was designed in the seventies at the Silesian University of Technology. The project was first implemented in the form of an electronic device. This device was used for many years to teach students the basics of the design and operation of computers. Currently, the W-Machine simulators are used for this purpose. The W-Machine is a didactic computer designed according to the von Neumann architecture. However, its basic version is devoid of IO devices and consists of the following components: arithmetic-logic unit, main memory, and control unit. These elements are connected by a data bus and an address bus. Fig. 1 shows the architecture of the W-Machine. Words representing the instructions and the data are usually sent on the data bus, and the instructions' addresses and data addresses are sent on the address bus.

The main memory unit (RAM) includes two registers: a data register and an address register. The address register contains the address of the memory location for which read and write operations are performed. The data register stores data that is written to and read from memory. Memory stores words that describe the instructions being executed and the data used by those instructions. The data is written as binary signed numbers in the two's complement system. The instruction description word is divided into two parts: the instruction code field and the instruction argument field. The argument of the instruction is most often the address in the memory. The arithmetic logic unit (ALU) performs calculations on the data delivered from memory. The result of the calculations is stored in a register called the accumulator.

The operation of the W-Machine is controlled by a control unit that generates appropriate control signals based on the current state of the machine. Control signals are binary signals that activate relevant operations performed in the processor. Each control signal is responsible for performing one elementary action. Table 1 shows the control signals and the actions they perform. The control unit also includes two registers: an instruction register and a program counter. The instruction register stores the description of the currently executed instruction.

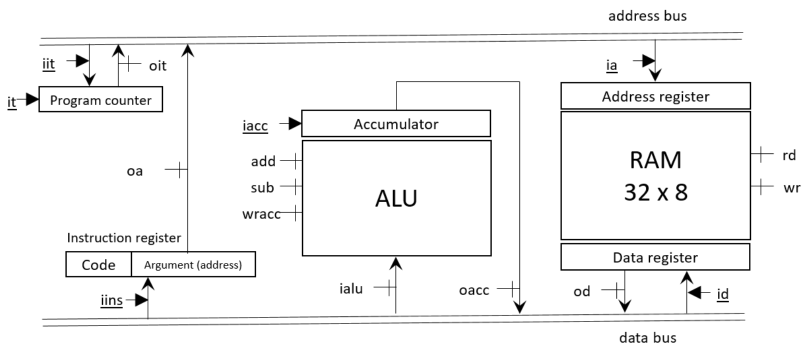


Fig. 1: W-Machine architecture.

Table 1: Control signals in the W-Machine

Signal name	Action
rd	reads a memory location addressed by the address register
wr	writes the content of the data register to the memory
ia	the content of the address bus are written to the memory address register
od	outputs the content of the data register to the data bus
id	writes the content of the data bus to the data register
ialu	the content of the data bus is fed to the ALU
add	computes the sum of the accumulator content and the ALU input
sub	subtracts the content of the ALU input from the accumulator
wracc	rewrites the content of the ALU input to the ALU output
iacc	writes the result of the operation in ALU to the accumulator
oacc	outputs the content of the accumulator to the data bus
iins	writes the content of the data bus to the instruction register
oa	outputs the address (the content of the argument field) to the address bus
oit	outputs the content of the program counter to the address bus
iit	writes the content of the address bus to the program counter
it	increments the content of the program counter

Table 2: Default W-Machine instruction list

Instruction name	Action
ADD	adds the memory word to accumulator
SUB	subtracts the memory word from accumulator
LOAD	loads the memory word to accumulator
STOR	stores the content of the accumulator to memory
JMP	jumps to the given address
BLZ	(branch on less than zero) branches when a negative number in accumulator
BEZ	(branch on equal to zero) branches when the accumulator is zero

Based on the content of this register, the control unit knows which instruction is currently being executed. The program counter contains the memory address where the currently executed instruction is located. However, due to the adopted concept of instruction execution, the program counter stores the address of the next instruction to be executed for most of the instruction execution time.

The execution of each instruction is divided into several successive cycles. The end of each cycle is indicated by a clock signal. Students should try to execute the instruction in as few cycles as possible. In each cycle, the corresponding control signals are activated, which causes the execution of the related actions. These actions constitute a single step of the instruction execution. Therefore, designing the instruction covers determining which control signals are to be active in each of the clock cycles. For example, the ADD instruction consists of three consecutive cycles. The signals *rd*, *od*, *iins* and *it* are active in the first (fetch and decode) cycle. The signals *oa* and *ia* are active in the second cycle. And finally, in the third cycle, they are active *rd*, *od*, *ialu*, *add*, *iacc*, *oit* and *ia*.

Table 2 shows the default list of instructions. Since the control unit is implemented as a microprogrammed one, we can modify existing instructions and add new ones. Moreover, students can use the designed instructions to write programs in a assembly language. With the W-Machine on the cloud, both functionalities can be practiced online, bearing in mind that it requires an active subscription for hosting the simulator operable and covering the hosting costs of the public cloud.

3 Related Works

Cloud computing is an architecture that works effectively in business. Although education has different priorities than business, these two areas have a lot in common. Just like in business, it is essential to reduce costs with high ease of use and flexibility of access. Of course, cloud computing is not intended for all applications. The research presented in [13] proves that in educational applications, high availability and quick adaptability to the changing demand for access are sometimes more important than performance.

The main goal of computer model simulators is to provide students with the opportunity to combine theory with practice. In [10], Prasad et al. analyzed over a dozen simulators for teaching computer organization and architecture, taking into account, i.a., the criteria, like scope, complexity, type of instruction set, possibility to write assembly code, user interface, support for distance learning and free availability. In conclusion, the authors stated that two simulators, MARIE and DEEDS[3] had met defined criteria. MARIE (Machine Architecture that is Really Intuitive and Easy) [9] is a simulator that has two versions: Java program and JavaScript version, and its main advantages are intuitive interface and assembly program execution visualization. DEEDS (Digital Electronics Education and Design Suite) covers combinational and sequential circuits, finite state machine design, computer architecture, and Deeds-McE (a computer emulator). The authors have found DEEDS to be less intuitive than MARIE, but its main advantage is the support for the generation of chip/PCB layout. Other tools analyzed in the article include Qucs (Quite Universal Circuit Simulator), CircuitLogix, ISE Design Suite – Xilinx, Quartus II, and HADES. In [6], Imai et al. present VisuSim used for assembly programming exercises in e-learning cooperative (built-in email communication) mode. Another worth mentioning simulator is the gem5 simulator - a very powerful, universal, and popular tool [8]. Nevertheless, gem5 is too complicated to use for first-year students with little knowledge of how computers work. The same problem applies to the Sim4edu website [4]. On the other hand, there exists a very simple online simulator[5], but it only allows for running assembly programs. In conclusion, none of the mentioned simulators allow students for simple, control signals-based definition of new instructions and utilization of the instructions in developed programs. Moreover, none of the tools is available for the cloud environment, like the W-Machine presented here.

[3] https://www.digitalelectronicsdeeds.com/deeds.html
[4] https://sim4edu.com/
[5] https://schweigi.github.io/assembler-simulator/

4 W-Machine on the Cloud Environment

The previously used W-Machine simulator was redeveloped for the Azure cloud. Cloud provides a hosting environment for scalable web applications, so it seems to be a good alternative for local web servers in the case of many (e.g., thousands) users. Moreover, the cloud environment lowers the entry barrier for hosting and ensures the high and global availability of the deployed applications, like the W-Machine. The tool has been divided into several individual cloud modules (Fig. 2). The bottom App Service is responsible for delivering the W-Machine application, which serves the website files using the HTTP protocol. It is the least loaded element of the system, sending files only the first time the W-Machine service is launched. Subsequent actions on the website are handled by servers with domain logic. The architecture includes two alternative server solutions responsible for the operation of the W-Machine. The first one is deployed using the Azure App Service (PaaS solution) with domain logic implementing the simulator's functionality. The second twin service runs on Azure Functions (serverless solution) with the same functionality. Both service approaches were chosen for their capability to scale an application easily (vertically and horizontally).

Both solutions use the same execution code supporting the logic of the W-Machine. The separation allows changing the front-end that cooperates with the simulator at any time of its operation. Additionally, each back-end service has its own Application Insights service responsible for monitoring its work. The Application Insights services are used to test the response time to requests sent in load tests presented in Section 5. The non-relational Redis database is responsible for storing user data, such as the state and settings of the W-Machine for each created session. The database features high efficiency thanks to storing data in a cache, which significantly speeds up the time of writing and reading data at the expense of data persistence and the maximum amount of information stored. The nature of the application does not require collecting a lot of data in a permanent form. Such architecture allows for effective separation of the website responsible for the user interface (which is unnecessary when testing the solution in terms of performance) from the computing services (implementing the teaching application logic), monitoring services, and databases.

The front-end of the W-Machine on the Cloud is shown in Fig. 3. The left panel allows observing automatic or manual execution of the program or instruction prepared by students. The right panel is used for developing the program (visible) or developing the instruction composed of control signals.

5 Experimental Evaluation of Alternative Architectures

For each alternative architecture of the execution environment, we carried out different types of performance tests to check the profitability of each architecture with possible scaling: 1) baseline, 2) load, 3) stress, 4) peak, and 5) soak testing. Each test was performed with the same test procedure. In the first step, we sent a request to get the state of the W-Machine, which simulated loading the

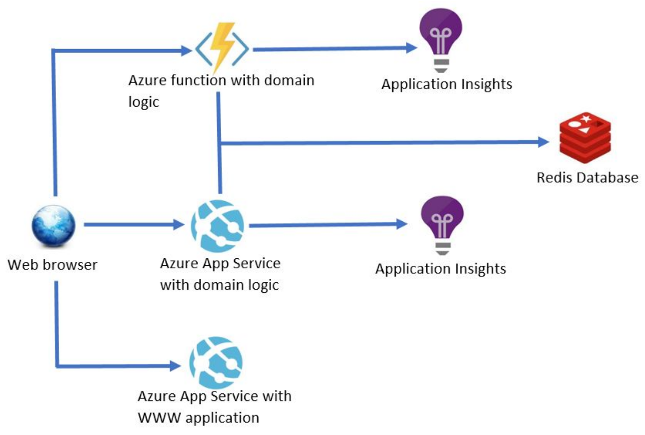

Azure function with domain logic
Application Insights
Redis Database
Web browser
Azure App Service with domain logic
Application Insights
Azure App Service with WWW application

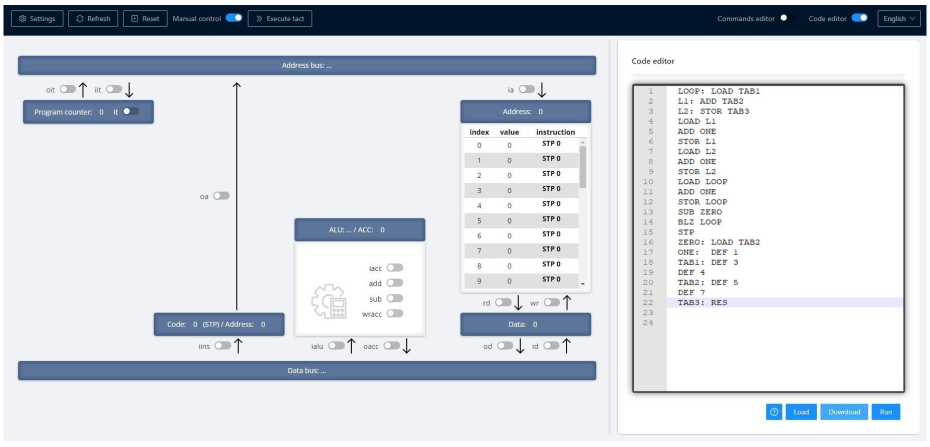
Fig. 2: Architectural alternatives of the W-Machine on the Cloud.

application for the first time. Then, we made a request to execute a simple program that summed two arrays and saved the result to the third array. The program used in all testing cases is the same as the one visible shown in Fig. 3. Tests were executed by repeating the simple program simultaneously and many times by multiple users. This was done by using K6 open source testing tool that simulates many users and sends execution requests to the tested environment. In our experiments, particular requests were sent with a 500 ms pause between one another since users usually do not act continuously, and a slight time delay accompanies each action they perform.

To test both investigated architectures (PaaS and serverless), the expected maximum load limit was set to 1,000 simultaneous users of the W-Machine. The successive load levels used in the tests were the percentage of the maximum load:

- baseline testing - 200 users (20% of the maximum load),
- load and soak testing - 700 users (70% of the maximum load),
- stress testing - $\leq$1,000 users most of the time, up to 1,500 users,
- peak testing - up to 1,500 users (150% of the maximum load).

The characteristics of performed tests are presented in Fig. 4. As can be observed, the baseline and load tests had three phases taking 1 minute, 3 minutes, and again 1 minute. In the first 1-minute phase, the number of users increased to the top value, appropriate to the test type. In the second phase, the number of users was constant, and in the third phase, it decreased again from the top value to zero. The difference between the tests lies in the top value. The nature of the soak test was similar except that it was much longer with 2-, 56-, and 2-minutes periods and 700 users. The stress test relied on the load constantly increasing to a level exceeding the maximum level that the system could achieve without generating a large number of errors. The expected place for a system breakdown was a level greater than 1,000. The peak test had a long period of



index	value	instruction
0	0	STP 0
1	0	STP 0
2	0	STP 0
3	0	STP 0
4	0	STP 0
5	0	STP 0
6	0	STP 0
7	0	STP 0
8	0	STP 0
9	0	STP 0

Code editor

```
LOOP: LOAD TAB1
L1: ADD TAB2
L2: STOR TAB3
LOAD L1
ADD ONE
STOR L1
LOAD L2
ADD ONE
STOR L2
LOAD LOOP
ADD ONE
STOR LOOP
SUB ZERO
BLZ LOOP
STP
ZERO: LOAD TAB2
ONE:  DEF 1
TAB1: DEF 3
DEF 4
TAB2: DEF 5
DEF 7
TAB3: RES
```

Fig. 3: The W-Machine on the Cloud: left panel for observing or manual testing of the program execution, right panel with a program or instruction editor.

the normal load of 200 users, followed by a sudden increase in the number of users to a critical number of 1,500 concurrent users, sustained for a short period. After that, traffic decreased drastically and went back to the previous level.

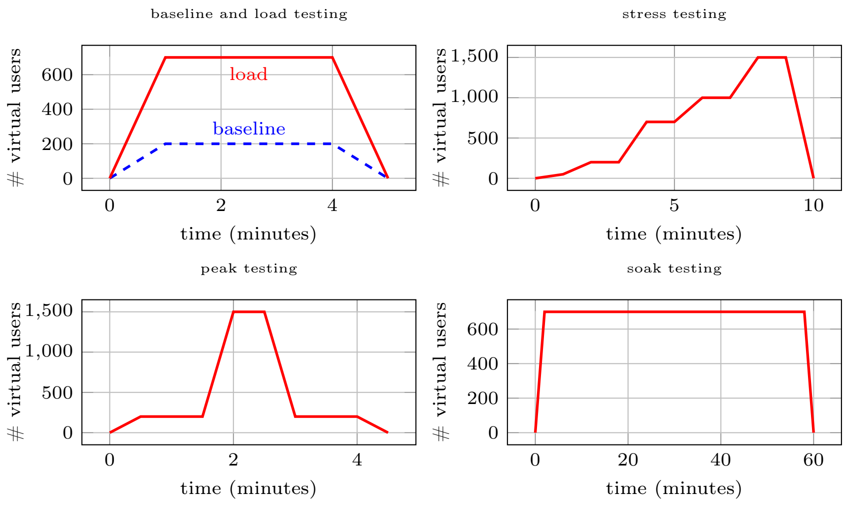


Fig. 4: Characteristics of the performed types of tests.

5.1 Performance Metrics

Each of the performed tests resulted in a collection of metrics, which we then used to analyze and compare performance against other system configurations. We used the following performance metrics to compare the alternative configurations and architectures of the W-Machine simulator:

– The number of requests per second (marked as $requests/s$) - the average number of requests sent throughout the test period. High values mean better application performance under heavy loads.
– The number of successfully completed requests - the sum of all requests successfully completed during the entire test.
– Number of requests with error status - the sum of all requests with errors.
– Success rate (marked as $OK(\%)$) - the percentage of successfully completed requests to all requests.
– Response time metrics: well-known average (marked as Avg), minimum (Min), maximum (Max), and median (Med), i.e., 50th percentile, as well as 90th percentile ($P(90)$) and 95th percentile ($P(95)$) which reflects the behavior of the system and its stability for sudden jumps in the application load (e.g., the value $P(90) = 1000ms$ means that 90% of all sent requests were completed in less than 1000ms, with 10% of all requests exceeding this time).

5.2 Results

For the W-Machine running on App Services (PaaS), the experiments were conducted separately on several pricing tiers with the varying number of instances of virtual machines (VMs) - 1, 3, up to 10, if it was possible for the particular tier. Such tests allowed us to select the best offer in terms of the target system performance and the costs of keeping it working. We started with the B1 tier, which provides the weakest virtual machines in terms of performance, and does not allow automatic scaling of the number of instances. Additionally, it has a maximum number of parallel applications of 3 at the same time. This tier is used mainly for development and testing. Results of various types of tests are shown in Table 3. As can be observed, the average response times (Avg) in various tests usually reach several seconds even when scaling to 3 instances. This shows that this tier does not guarantee appropriate performance.

The B3 pricing tier, compared to B1, offers 4 times more computing power, which translates into much better results (Table 3). The median (Med) and average response times for baseline testing are well below 100 ms. The obtained results are almost 10 times better compared to the B1 pricing tier. Bandwidth has also increased drastically ($requests/s$). Adding another 2 instances seems to be necessary only for more heavy loads. For example, for the stress test we can observe that the value for P(95) decreased two-fold to a satisfactory level after scaling from 1 to 3 instances. The cost of one B3 instance is 30% higher than that generated by three B1 instances, but the results obtained are many times better. The instances in the S1 tier have comparable parameters to those

Table 3: Performance tests for W-Machine working as an App Service for different pricing tiers and variable instance count.

pricing tier	testing type	insta-nces	cost	requ-ests/s	OK(%)	Response time to request (ms)					
						Avg	Min	Max	Med	P(90)	P(95)
B1	baseline	1	54.75	94.14	100	1210	43	5210	1110	2470	2790
		3	164.25	193.82	100	324.33	40.83	3780	158.07	845.85	1150
	load	1	54.75	90.43	99.85	5890	42.52	22850	6180	9060	10090
		3	164.25	223.88	100	2030	41.23	21530	900.5	5970	7830
	stress	1	54.75	84.2	98.53	8010	42.41	60000	6640	16650	20050
		3	164.25	220.42	100	2740	40.65	29280	788.54	8970	11270
	peak	1	54.75	88.92	100	5850	43	45170	2090	19260	26560
		3	164.25	193.48	100	2440	41.09	45920	314.43	5290	17460
B3	baseline	1	219	271	100	86.82	40.44	839.8	71.1	152.92	192.61
		3	657	289.79	100	50.78	40.51	442.5	45.39	73.49	76.37
	load	1	219	447.44	100	754.29	40.96	3550	697.09	1430	1740
		3	657	868.12	100	142	40.22	1930	75.54	340.37	496.8
	stress	1	219	397.84	100	1240	41.04	6780	1060	2590	3280
		3	657	773.42	100	389.27	40.45	6650	111	1030	1720
	peak	1	219	340.84	100	925.12	40.7	5390	243.4	2710	3130
		3	657	516.76	100	422.23	40	5840	69.7	1150	2300
S1	baseline	1	73	102	100	1060	42	6540	971	2030	2370
		3	219	202	100	289	41	3800	132.75	736	1110
		10	730	277	100	75.86	40.53	1470	51.28	116.22	159.85
	load	1	73	105	100	4940	42	17060	5270	7640	8540
		3	219	228	100	1980	40.8	24100	886	5960	9270
		10	730	670.92	100	333.1	40.37	9580	128	744.91	1290
	stress	1	73	93	98.66	7080	42	60000	6110	14580	16910
		3	219	209	99.74	2910	41	38890	973.32	8910	11940
		10	730	537.67	100	722.53	40	22210	132.4	1840	3860
	peak	1	73	85	96.71	5940	43	60000	2110	13180	16700
		3	219	206	100	2230	40.98	35770	346.6	5010	15970
		10	730	444.12	100	608.86	39.74	12620	88.22	1540	4100
S3	baseline	1	292	258.16	100	115.84	40.46	2790	86.12	229.55	291.29
		3	876	288.91	100	52.44	40.39	1540	46.21	76.29	80.52
		10	2920	289.48	100	51.22	40.6	437.34	45.73	74.68	78.01
	load	1	292	427	100	812	40	4120	797	1510	1710
		3	876	810.14	100	188.09	40.22	2650	80.43	481.87	737.51
		10	2920	1012.85	100	51.76	40.21	477.4	45.88	75.15	79.3
	stress	1	292	377	100	1330	40	6370	1120	2850	3190
		3	876	718.28	100	459.4	40.31	6600	113.57	1390	2180
		10	2920	1229.77	100	56.7	40.27	1310	48.19	80.61	89.02
	peak	1	292	333	100	967	40	5630	235	2830	3300
		3	876	515.59	100	430.79	40.32	8200	71.2	1170	2210
		10	2920	837.77	100	59.95	40.03	813.87	48.43	83.49	98.2
	soak	3	876	970.88	100	214.31	39.16	3600	94.99	563.65	793.94
P3V3	baseline	1	957.76	291.5	100	46.92	40.17	1150	44.46	55.27	56.95
	load	1	957.76	1007.66	100	53.63	39.85	1380	46.3	68.77	82.63
	stress	1	957.76	1012.8	100	177.03	39.9	2470	94.92	439.7	556.21
	peak	1	957.76	717.69	100	157.71	40.08	1980	87.23	376.39	472.05

offered in the B1 plan. However, in the S1 tier, we can scale the application up to 10 instances, which translates into much greater available performance. In fact, single instances in this plan are slightly better than those available in the B1 plan. Increasing the number of instances to 10 brings noticeable performance gain for more demanding tests - for the load and stress tests the average response time decreased at least 10 times and the throughput increased at least 6 times.

The S3 tier offers performance identical to that available in the B3 plan. The main difference lies in the maximum number of instances that can be allocated to serve the application. Given the large amount of resources offered in this tier, the differences in the baseline test results for the varying instance numbers are negligible. The first differences are visible when the number of users is increased to 700 (S3, load testing). For 10 instances, the application works visibly below its maximum capabilities and still has a computational reserve, which can be seen in a slight increase in bandwidth ($requests/s$) compared to the configuration using 3 instances. Median and average response times are much lower than for the S1 tier and comparable to those obtained with the B3 tier for the same number of instances. The performed stress and peak tests also showed a large reserve of computing power available with the S3 tier and 10 instances. This can be noticed by observing the P(95) value, which remains at a level lower than 100ms. For this pricing tier (S3), we also carried out the soak tests using 3 application instances (S3, soak). After a 4-hour test, there were no errors in the application ($OK(\%){=}100$), and its operation during this time was stable (the number of requests per second was equal to 970.88 with the average response time 214.31ms). The achieved throughput was similar to that obtained in the load test for the same configuration (3 instances). The relatively low P(95) value of less than 1 second also proves that the load is distributed fairly evenly over the available application instances. Due to the high costs of the P3V3 pricing tier, we only tested the application working on one active instance. This layer provides extensive computing resources (8 cores and 32GB memory per instance). Comparing the results obtained for the P3V3-based deployment of W-Machine with those obtained for the configuration with the use of the S3 plan, we can notice how important the performance of a single machine is. The results obtained for the load test are almost identical when comparing them with the S3-hosted application with 10 instances, while the costs generated by both solutions are entirely different. The P3V3 pricing tier is about 3 times cheaper here, offering similar performance. Table 3 (P3V3) shows that the W-Machine simulator provides perfect performance results for all tests, keeping the median response time below the 100ms limit.

The application responsible for the main logic of the W-Machine uses the .NET 5 platform, which allows it to be implemented as an App Service using three different environments: Windows and Linux servers and the Docker containerization tool. The monthly costs generated by each of them are \$730, \$693.5, and \$693.5 respectively for Windows, Linux, and Docker (with the S1 tier configured for 10 application instances). To compare their actual performance, we performed the same tests for each OS platform. When testing the four different scenarios shown in Table 4, no significant differences in the results obtained were observed. The differences are within the limits of the measurement error and are not a sufficient basis for determining the superiority of a given OS platform, taking into account only the differences in performance. However, costs may vary significantly, especially for premium tiers, like P3V3 (\$957.76 per month for Windows, \$490.56 per month for Linux).

The second of the tested architectures for the W-Machine relied on serverless Azure Function service. This approach differs in operation from those using dedicated virtual machines. The first difference is the capability to scale active instances to zero. This results in a cold start of the application, which occurs when the website is inactive for a long time. Lack of network traffic deactivates all active instances executing the W-Machine code, so the first requests after resuming traffic take much longer. The second difference is the inability to set a fixed number of active instances, as we can do in App Service (PaaS). Automatic scaling of the application, which is an immanent feature of the approach, works very quickly, as indicated by very low P(95) values for the performed tests in Table 5 (we performed the same tests as for the various App Service configurations).

We could observe that in baseline testing, after just the first 30 seconds, 4 instances of the application were created, which remained until the end of the test. In load testing, 9 instances were allocated almost immediately that served 700 simulated users very efficiently. However, the maximum response time for the test exceeded 10 seconds due to the aforementioned cold start. Requests sent before creating the instances responding to the traffic adequately often resulted in a long delay in execution. A similar effect could be seen for stress testing and peak testing, where the increase in traffic is sudden and requires the creation of new instances of the application. The load spikes were much more pronounced here, and the first errors appeared due to the lack of computing power of the available instances. The stress testing started with 4 instances at the level of 200 simulated users. As the workload increased, the number of instances increased to 6 for 700 users, 10 instances for 1,000 users, and up to 14 instances for 1,500

Table 4: Performance tests for App Service on S1 tier for different environments.

testing type	environment	requests/s	OK(%)	Response time to request (ms)					
				Avg	Min	Max	Med	P(90)	P(95)
baseline	Windows	277	100	75.86	40.53	1470	51.28	116.22	159.85
	Linux	277.18	100	76.2	49.22	2060	51.16	118.55	155.25
	Docker	278.28	100	73.62	40.44	1630	50.28	114.77	156.01
load	Windows	670.92	100	333.1	40.37	9580	128	744.91	1290
	Linux	642.57	100	369.93	40.31	1319	115.69	918.4	1560
	Docker	631.29	100	385.66	40.49	10920	114.96	1000	1710
stress	Windows	537.67	100	722.53	40	22210	132.4	1840	3860
	Linux	539.2	100	784.23	40.27	19980	102.47	1230	5410
	Docker	563.4	100	727.91	40.54	18000	137.21	1730	3220
peak	Windows	444.12	100	608.86	39.74	12620	88.22	1540	4100
	Linux	435.82	100	621.29	40.35	15500	89.49	1430	2850
	Docker	452.92	100	568.02	40.47	12040	86.62	1650	3260

Table 5: Performance tests for W-Machine working on serverless Azure Function.

testing types	requests/s	OK(%)	Response time to request (ms)					
			Avg	Min	Max	Med	P(90)	P(95)
baseline	283.04	100	63.24	43.21	346	56.81	81.44	92.96
load	949.21	100	88.04	42.6	10150	69.68	140.71	186.73
stress	1063.51	99.99	144.06	42.97	56110	113.53	236.19	290.47
peak	665.99	99.95	185.24	43.38	19400	88.5	434.46	567.57

users. The dynamic addition of resources allowed the application to maintain a stable response time. The values of P(90) and P(95) did not exceed the limit of 300ms, which proves the very good responsiveness of the system. The biggest challenge for the application was the peak testing, which subjected the service to a very rapid increase in load. It took a full minute for the traffic to stabilize, during which the number of active application instances increased from 4 to 14.

6 Results Summary and Concluding Remarks

Providing computer simulators that can be quickly deployed, globally available in many regions worldwide, and fast responding to users' requests is crucial for globally implemented distance learning. In terms of functionality, our W-Machine simulator complements the solutions mentioned in Section 3 by providing a low entry barrier for observing and planning transfers of data and addresses, managing particular steps of the instruction execution (on the control signals level), and development and debugging of created instructions and programs implemented in assembly language. When surveying the usability of the tool among 71 first-year students, we obtained 73% satisfaction in terms of usability, 71% in terms of creativity and 51% in terms of the attractiveness of the tool. Moreover, to our best knowledge, it is the first cloud-based computer simulator. Thus, it is very flexible in the context of worldwide accessibility for a large group of users at the same time. The results obtained during our experiments allow formulating conclusions regarding the impact of individual parameters of the offered approaches on the actual system efficiency.

When starting the research on alternative approaches to the application deployment (App Services and Azure Functions), the primary assumption was to find the best solution in terms of performance and generated costs. To qualify the configuration as meeting the stable and quick operation criteria, we assumed the maximum median and average response time limit not exceeding 200 ms. Among all the tested versions, we can distinguish three candidate solutions. The first is a 3-instance B3 pricing tier setup, which translates into a monthly cost of $657.00. The price is for the Windows version; therefore, the final price may be reduced to $153.30 when using the Linux version. Unfortunately, this tier has limited scaling capabilities. Another configuration worth a closer look at is the one based on the S3 tier, which is 33% more expensive than B3 but also more scalable. The application running on the Azure Function service is also a promising alternative. In some cases, the best request throughput and average response times may also prove to be the best budgetary solution. For example, in the case of applications with short but intense periods of high load, a much better alternative is to choose the serverless service (Azure Function). However, with the assumed constant load, the situation may also change in favor of App Services with an appropriate pricing tier for the Application Service Plan.

Future works will cover further development of the W-Machine on the cloud covering the implementation of other architectures of the simulator itself toward the inclusion of the interrupt and I/O modules.

Acknowledgments

The research was supported by Statutory Research funds of the Department of Applied Informatics, Silesian University of Technology, Gliwice, Poland (grant no. 02/100/BK_22/0017).

References

1. Al-Mawee, W., Morgan-Kwayu, K., Gharaibeh, T.: Student's perspective on distance learning during covid-19 pandemic: A case study of western michigan university. International Journal of Educational Research Open ISSN 2666-3740 (2021)
2. Alashhab, Z.R., Anbar, M., Singh, M.M., Leau, Y.B., Al-Sai, Z.A., Alhayja'a, S.A.: Impact of coronavirus pandemic crisis on technologies and cloud computing applications. Journal of Electronic Science and Technology 19(1), 100059 (2021)
3. Encalada, W.L., Sequera, J.L.C.: Model to implement virtual computing labs via cloud computing services. Symmetry 9(7), 117 (2017)
4. Goldstine, H.H.: The computer from Pascal to von Neumann. Princeton University Press (2008)
5. Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach. Sixth Edition, Elsevier (2019)
6. Imai, Y., Imai, M., Moritoh, Y.: Evaluation of visual computer simulator for computer architecture education. In: IADIS International Conference e-Learning. pp. 17–24 (2013)
7. Jararweh, Y., Alshara, Z., Jarrah, M., Kharbutli, M., Alsaleh, M.N.: Teachcloud: a cloud computing educational toolkit. International Journal of Cloud Computing 1 2(2-3), 237–257 (2013)
8. Lowe-Power, J., Ahmad, A., Armejach, A., Herrera, A., , et al.: The gem5 simulator: Version 20.0+. (2021), https://hal.inria.fr/hal-03100818/file/main.pdf
9. Nyugen, J., Joshi, S., Jiang, E.: Introduction to MARIE, A Basic CPU Simulator. The MIT (2016)
10. Prasad, P., Alsadoon, A., Beg, A., Chan, A.: Using simulators for teaching computer organization and architecture. Comput. Appl. Eng. Educ. 24(2), 215–224 (2016)
11. Soni, V.D.: Global impact of e-learning during covid 19. Available at SSRN 3630073 (2020), https://dx.doi.org/10.2139/ssrn.3630073
12. Stallings, W.: Computer Organization and architecture. Designing for Performance. Pearson, Hoboken, NJ, 10 edn. (2016)
13. Tamara Almarabeh, Y.K.M.: Cloud computing of e-learning. Modern Applied Science ISSN 1913-1844 pp. 11–18 (2018)
14. Tanenbaum, A.: Structured Computer Organization. Pearson, Uppersaddle River, NJ (2013)
15. Wang, B., Xing, H.: The application of cloud computing in education informatization. In: 2011 International Conference on Computer Science and Service System (CSSS). pp. 2673–2676. IEEE (2011)
16. Węgrzyn, S.: Foundations of Computer Science. PWN, Warsaw (1982)
17. Yadav, K.: Role of cloud computing in education. Int. Journal of Innovative Research in Computer and Communication Engineering 2(2), 3108–3112 (2014)