International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Cloud Deployment Effectiveness and Data Security

Dr. Md Dilshad Ghani

Magadh University Bodh-Gaya Bihar India
Email: *dislhadghani[at]hotmail.com*

Abstract: ***Software service providers are increasingly taking over the cloud computing paradigm to provide on-demand access to near-unlimited resource pools. Services, memory sensitivity Cloud data and rationalization. Two of the central issues that concern the industry: (i) how to achieve outsourced calculations in an efficient and manageable way through agile and optimized deployments, and (ii) Especially with sensitive data. How sensitive data is not protected public cloud, but still maintains business functionality practically. To optimize deployment, cloud providers have improved their computing infrastructure. By introducing container orchestration, we introduce the framework with improved software proposal automation and agility. Additionally, software service providers have migrated computationally intensive software systems such as high layers and technical workflows to achieve efficiency. To maintain the benefits of cloud-protected data, researchers have proposed many encryption techniques for searching and computation of encrypted data. However, software practitioners face several challenges. First, elastic scaling of arithmetic resources such as containers leads to non-negative delays. This is called cold start and prohibits the agile and rapid delivery of software services. This concern is often the case that server less computing and generally automated scaling systems (deadline-based service level goals (SLOs) for deadline-based applications. There are many techniques for resource instantiation. The problem is that cloud resources can be reduced to minutes. The contribution is based on several applications in the fields of aviation, finance and healthcare. The results were verified and evaluated. This paper shows how to effectively tackle the above challenges. Finally, the paper outlines future directions beyond the limits of success presented.***

Keywords: HPC clusters, EWIS, MDO, MAPE-K loop

1. Introduction

Using optimization processes to develop, model, and refine intricate designs. In order to simulate and optimize physical attributes like strength, vibrations, geometrical decomposition, or material selection, these workflows are intricate and time-consuming procedures that are usually made up of a variety of software tools and services. These tools can include engineering tools like MathWorks, Cradle, and others, as well as knowledge engineering tools like ParaPy. To carry out these operations, engineers utilize a variety of hardware, such as their desktop PCs or High-Performance Computing (HPC) clusters.

Current circumstances. The processing, memory, and storage capacities of desktop computers are con-strained. Furthermore, the experiments' parallel execution is dependent on the quantity of computers and cores that are accessible. Although HPC clusters are more powerful and efficient than desktop computers, they are built with specialized, costly gear, and their capacity isn't always immediately available. For engineers who conduct ongoing or repeating experiments, scheduling time slots and complicated queuing APIs present still another challenge.

The cloud's promise. These days, engineers can use cloud computing to obtain on-demand access to the resources they need for their processes, which are frequently built on inexpensive commodity hard-ware. A common pool of reconfigurable computing resources, such as networks, servers, storage, apps, and services, can be accessed on-demand using the cloud computing concept [14]. Automated provisioning of necessary cloud-related resources, including virtual networks, virtual machines, and necessary infrastructure software and middleware platforms, is made possible by cloud orchestration solutions. As a result, deployment procedures and infrastructure are now fully programmable and decomposable.

2. Challenges

When implementing and carrying out engineering workflows in the cloud, there are still significant issues and difficulties. In addition to supporting intelligent scaling and the execution of simulation and optimization workflows in the cloud, engineers must automate deployment. This procedure uses adaptive deployment for every workflow deployment and execution in order to group, divide, and parallel-size the various jobs and their corresponding tools on the appropriate number and kind of nodes. Both may also change based on the particulars of a certain execution. Automating this process involves automatically estimating the quantity of cloud resources (number of virtual machines, memory, and cores, etc.), automatically determining the types of resources needed (virtual machines, storage volumes, etc.), and automatically bootstrapping and destroying the necessary infrastructure.

InfraComposer: towards smart deployment in the cloud.
In order to overcome these obstacles, we introduce a reflective and flexible middleware that facilitates and controls (i) the deployment of intelligent, flexible workflows, (ii) scaling, and (iii) cloud execution. We make use of both domain-specific expertise on the specific tools utilized and thorough examination of these tools when they are implemented and run on the cloud platform.

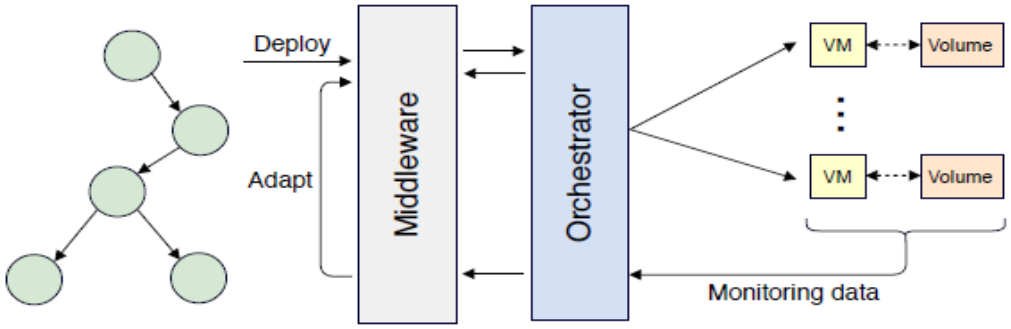

Figure 1.1: Middleware for cloudifying simulation workflows

The engineers' feedback regarding the characteristics of the tools they use and their execution history both influence the adaptive middleware. In terms of CPU, memory, and network utilization, the engineers' contribution is defined as annotations on the processes and is predicated on human expertise and assumptions about the tools. The original deployment and scaling strategy will be optimized using the execution history over time, allowing for further adaptation to actual execution information (see Fig. 1.1).

Therefore, our middleware identifies two significant additions to the current deployment and orchestration middleware:

1) Resource deployment and reservation planning based on annotations. An initial deployment plan will be created using the workflow's annotations. A cloud orchestrator can carry out a deployment plan, which is a topology and orchestration specification for a cloud application.
2) Adaptive scaling and reconfiguration based on history. The middleware uses past outcomes to modify the deployment plan configurations according to the workflows' execution history. This is fuelled by policies that have the ability to analyse execution history statistically and reason about it.

In order to do this, the middleware's reflective meta-models allow for the reification of:

1) Important architectural ideas like tasks, workflows, particular tools, and how they are deployed,
2) Important ideas about how certain tools should be executed with particular parameters; and
3) Important ideas about how resources should be used, like nodes, cores, CPU time, memory, storage, and network metrics.

Electrical Wiring Interconnection System (EWIS):-

EWIS is a crucial phase in the multidisciplinary design optimizations (MDO) process used in aircraft design. To determine the best options for cockpit wire harness routings, aerospace experts plan and carry out intricate simulation and optimization tests.

Motivating Situation. It is unclear whether a task's primary optimization tool is memory-intensive when looking at the tasks in an EWIS workflow. The best deployment is accomplished through a gradual and flexible procedure that is initially influenced by the engineer's annotations and subsequently by the tool's execution history.

1) Because the wire harness tool puts a lot of data, such as an aircraft's physical characteristics, into memory, the aerospace engineer specifies that it is memory-intensive rather than CPU-intensive. This is, at the very least, the engineer's annotation-based assumption.
2) The middleware allows the engineer to carry out the experiment as outlined in the process by allocating a significant quantity of RAM with a limited number of cores for every virtual node in the cloud.
3) However, because the executing nodes surpass the system load saturation limits and the memory resource has been over allocated, the execution history indicates that the tool is mostly CPU intensive.
4) The deployment strategy ought to be modified to use more cores and less memory for the virtual nodes.

Design of the hinge system of an aircraft rudder:-

The design of airplanes is another instance of multidisciplinary design optimization (MDO). An auto-mated workflow optimizes the hinge design, which is an essential component of the overall rudder de-sign of an airplane. In order to minimize engineering goals (such as overall weight), the workflow in-clues a collection of engineering tools that handle meshing and stress analysis of hinge components in addition to doing a nearly thorough search for every potential solution [97, 114]. As a validation, we provide a workflow in Section 3.5.2.

Motivating scenario. The workflow's execution flow for these interdependent engineering tools is predicated on intricate designs created by MDO specialists. As a result, an aeronautical engineer cannot immediately determine the number of nodes, the necessary tool instances, or the supported level of parallelization.

1) To begin, the workflow engineer indicates through annotations that only a very large node is needed additionally, an anticipated execution completion time is specified by the engineer.
2) In accordance with the procedure, the middleware instantiates the node and carries out the experiment.
3) How-ever, due to a huge number of sub-experiments, the execution history reveals that the execution takes a lot longer than anticipated. This leads to numerous scheduled jobs and parallel runs.
4) In light of the anticipated execution time, the deployment plan should be revised to modify the number of nodes (rescaling).

Electrical Wiring Interconnection System Design

The wire harness routing simulation and optimization procedure is demonstrated by the condensed EWIS workflow (see Fig. 1.2). Usually, a workflow engine (Optimus in our instance [185]) powers such a workflow. Such optimization

procedures involve concurrent, iterative execution of tests to pro-vide the best wire routing. Different input design variables are supplied by the workflow engine at runtime for every experiment run. The creation, coordination, and offloading of jobs to worker nodes are handled by the workflow engine.

Following the MAPE-K loop's three distinct iterations (steps), we conducted a series of tests that presented various adaption scenarios. In each scenario, we suppose that an engineer has made a series of incorrect assumptions about the workflow and its requirements. The MAPE-K loop is iterated in each step.

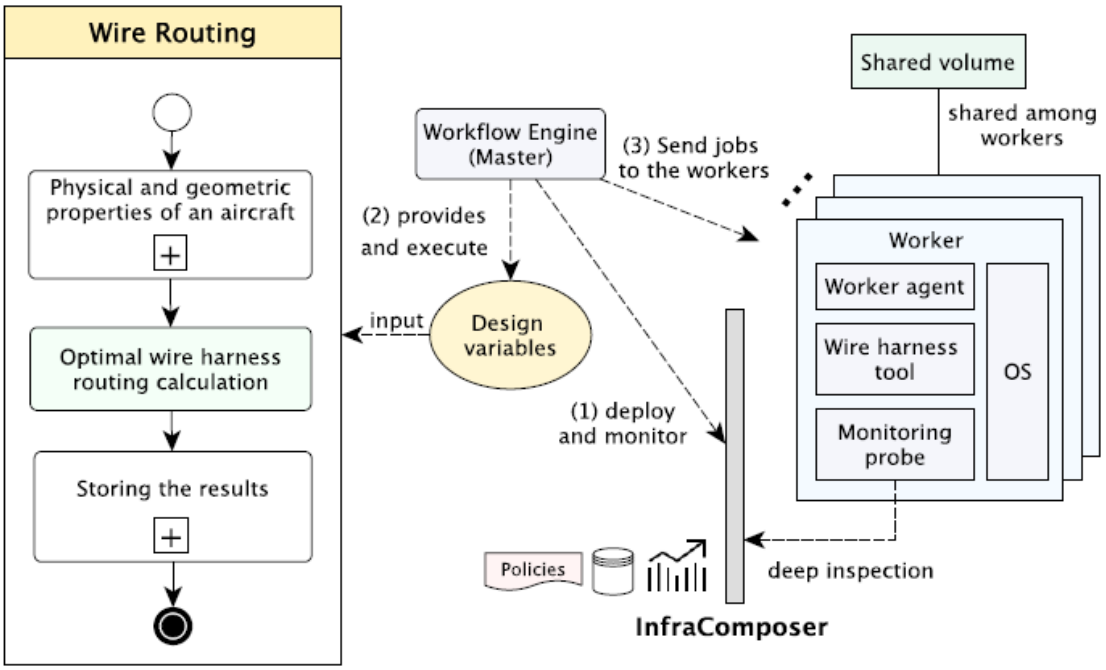

Figure 1.2: An overview of the cloud-based wire harness routing simulation and optimization process is shown in Both the worker nodes and the workflow engine (master node) are set up on virtual computers.

Table 1.1: Configuration details of each execution round of the EWIS workflow.

	vCPU	RAM (GB)	Nodes	Intensiveness	Execution	Jobs per VM
Step 1	2	16	5	Memory Network	62.37 min	9
Step 2	4	8	10	CPU Network	33.10 min	~5
Step 3	4	4	15	CPU Network	22.02 min	3

Scenario 1: the workflow is memory intensive: The reflective monitoring indicates that the wire harness tool does not load a significant amount of data into the memory before or during the workflow execution, despite the engineer's annotation that it is memory hungry. As a result, the deployment strategy is modified to use virtual machines with less RAM and to scale down.

The workflow engineer specifies annotations for resource use and direct deployment in phase 1. The former consists of five worker nodes, one installation per node, the usage of the wire harness optimization tool, and a shared storage space for geometrical data. The latter characterizes the process as requiring a lot of memory and network. The cloud infrastructure is then deployed by sending the work-flow to Infra Composer. The execution then begins, and the workflow engine (the master node) for-wards optimization jobs to the worker nodes, as shown in Fig. 1.2. regulations. Every execution step is a 45-experiment iteration of the MAPE-K loop. In this use case, each node can accept one work (optimal wire harness routing computation) at a time. The number of experiments is specific domain knowledge, and InraComposer observes the 45 experiments as one large experiment.

The engineer's next goal is to repeat the process with other experiments. Following a thorough examination of various jobs, processes, and system-wide KPIs, InfraComposer continued to monitor the execution of step 1.

To enforce the business rules, the monitoring component collects memory-related metrics (such as free, swap memory, etc.) and sends the results to the policy engine (see Table 3.3). In step 1, a significant amount of memory was left unused, as seen in Fig. 3.9. To prevent over-provisioning, the virtual machine has been shrunk to use less memory.

S	Adaptations	Policies
1	-Assumption: memory intensive – Adaptation: scale up/down the VM	input :VM if VM.free_memory > 4GB and VM.total_memory >= 4GB then | resize(VM, VM.total_memory/2) if VM.swap_memory > 0 then | resize(VM, VM.total_memory*2)

2	--Assumption: CPU intensive – Adaptation: scale up/down the VM	input :VM if VM.load_average < VM.cores and VM.CPU_utilization < 40% and VM.cores >= 4 then | resize(VM, VM.cores/2) if VM.load_average > VM.cores or VM.CPU_utilization > 60% then | resize(VM, VM.cores*2)
3	– Assumption: few number of nodes – Adaptation: scale in/out the VMs	input : Cluster, VM, Workflow, Net, Master, license, quota if (Workflow.execution > 30min or VM.accepted jobs > 1) and Net.bandwidth_utilization < 1Gbps and Net.packet_loss < 5% and is_sufficient (license, quota) and not master_node_saturated(Master) then | quantity license.tools < 5 ? license.tools : 5 | resize(Cluster, Cluster.VMs.count + quantity) if Net.bandwidth_utilization >= 1Gbps or Net.packet_loss >= 5% or master_node_saturated(Master) then | resize(Cluster, Cluster.VMs.count - 5)

Table 1.2: Policy-based adaptation scenarios based on execution history. The system performance analysis is performed using the Utilization Saturation and Errors (USE) [82] methodology. Aggregated values are maximum values, and VMs. Load average maps to the system load during the last one-minute periods in Linux.

Scenario 2: the workflow is not CPU intensive Although the reflective monitoring data indicates that the nodes and the tools utilize more than a specific threshold, the workflow—specifically, the wire harness optimization task—is not tagged as compute-bound. More precisely, when the load number is greater than the number of cores, the virtual machine load average illustrates system saturation. In order to scale up and use more cores, the deployment plan has been changed.

Following step 1, InfraComposer uses its policy engine to enforce a few compute-related rules in order to re-execute the workflow (see scenario 2 in Table 3.3). The CPU utilization and system load over the last one-minute intervals (average Linux load of one minute) are shown in Fig. 3.10. System saturation is indicated by the Linux system load, which should typically be lower than the number of cores. The system is saturated since the virtual machine used two cores in step 1. As a result, the workflow is changed to use four cores for steps two and three. The system load thus stayed below 4 (the number of cores).

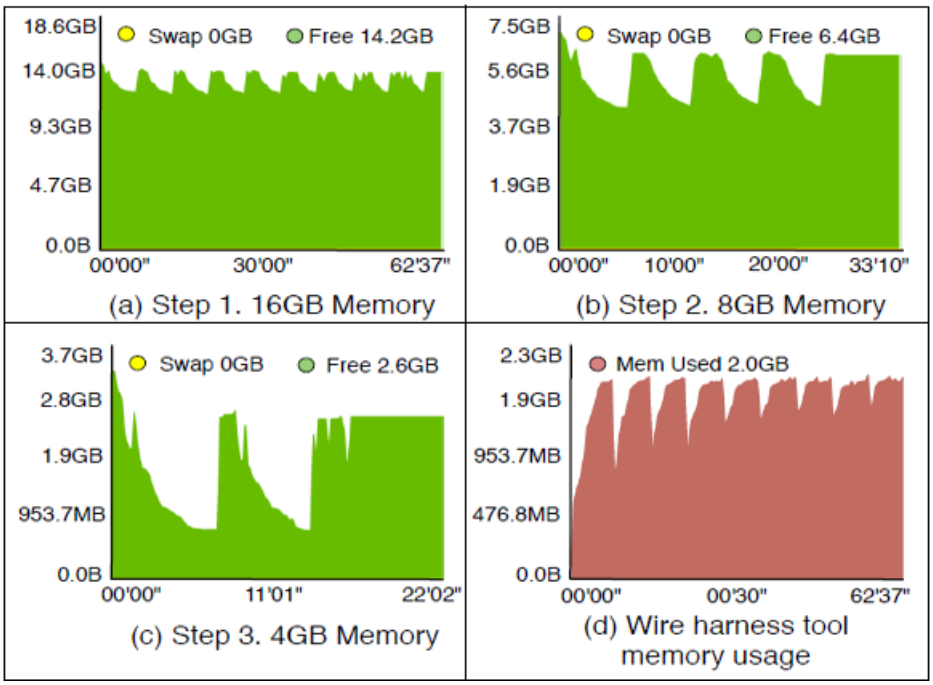

Figure 1.3: Different steps of the EWIS case: free memory vs. swapping

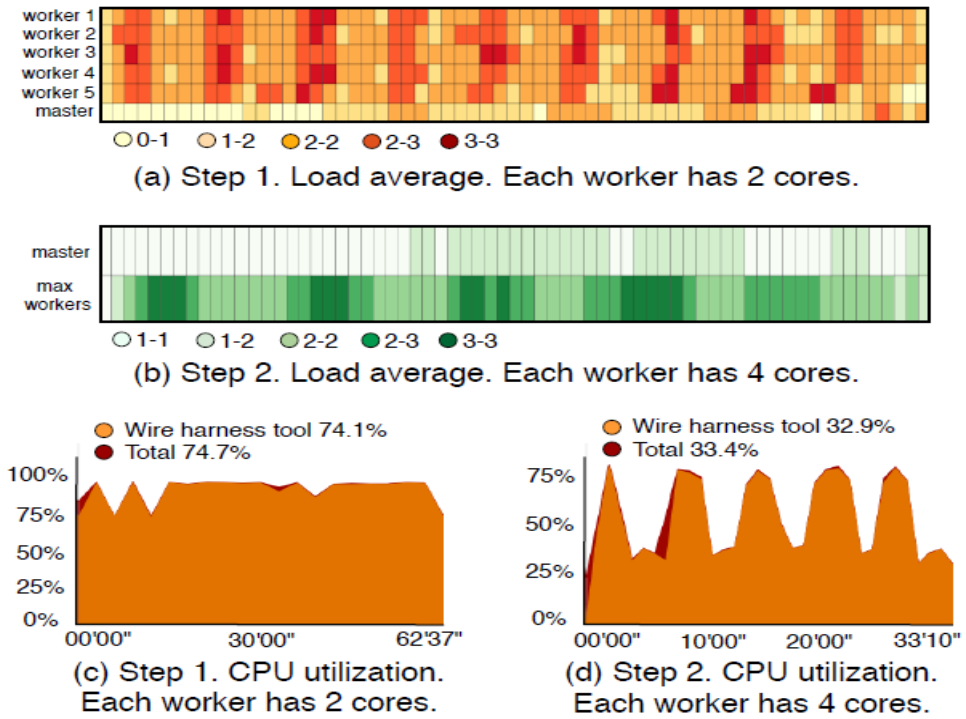

Figure 1.4: CPU utilization and load average of the workers and the master node. Master node has 4 cores. Each core is a virtual core.

Scenario 3: the workflow needs few number of nodes

The engineer expects the wire harness virtual machines to have a restricted horizontal scale. In the distributed arrangement, he must adhere to his quota of total cores and memory, and at first, he wants it to be inexpensively installed on a small number of instances. However, because of the quantity of the input parameters and datasets (about the physical attributes of the aircraft), reflective monitoring reveals that there are a significant number of parallel executions (i.e., scheduled jobs). As a result, the total execution is prolonged. As a result, the deployment strategy is modified to reduce the number of nodes and their memory capacity. Because of this, the process can use more instances of the tools, meeting the predicted execution time need while adhering to budget and quota.

Horizontal re-scaling limits are presented as rules in policies in Table 3.3 for scenario 3. Networking metrics, licensing, and quota restrictions are also crucial, in addition to execution time and the quantity of approved jobs per virtual machine (VM) (9 jobs per VM in step 1). There is a real upper limit to cluster-wide bandwidth consumption, and beyond it results in network saturation in terms of packet loss, networking failures, and segment retransmissions. Additionally, because of the job scheduling and workflow execution, resources ought to be available in the master node. Policies ensure that disk IO speeds, available RAM, and average load do not overload the master node.

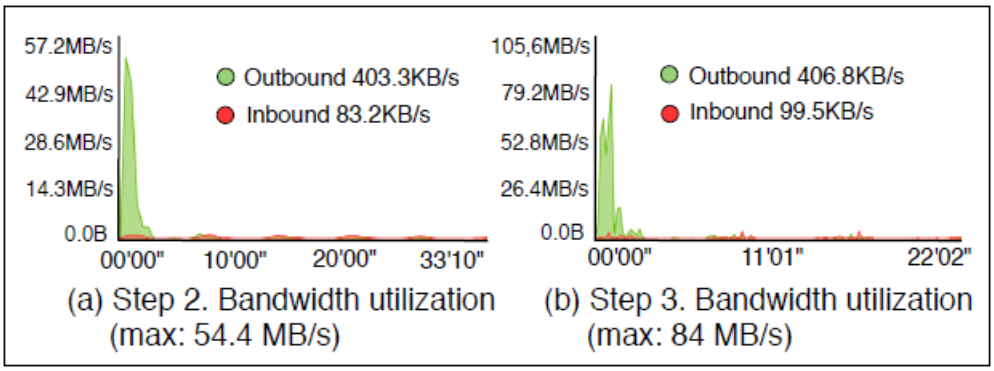

Figure 1.4: Cluster-wide bandwidth utilization of the network where the shared volume is operating.

The network's cluster-wide bandwidth usage when the shared volume is in use is shown in Fig. 1.4.

The shared volume is in use. Packet loss was minimal because the maximum incoming rate (34.9 MB/s in step 1 (not shown), 54.5 MB/s in step 2, and 84 MB/s in step 3) is less than the network's total bandwidth. Furthermore, the nodes' disk IO utilization was 52.4% in step 1, 74.82% in step 2, and 70.05% in step 3. These numbers are affected by the shared volume's network latency. When compared to step 1, the execution time was 29.27 minutes faster in step 2 and 40.25 minutes faster in step 3 as a result of the policy engine scaling out and reconfiguring the number of nodes to 10 in step 2 and 15 in step 3.

Limitations and Future Opportunities
This section describes our design choices and future directions for (i) componentization granularity and (ii) the policy-based decision making mechanism in the context of cloudification engineering work-flows.

Granularity of Componentization Although virtual machines are provided by InfraComposer as the level of componentization, the ideas in the architecture are not always associated with any particular component model (such as virtual machines or containers). Applying the ideas discussed in this work to a containerized environment, which calls for minimal concerns, is part of our future work. (i) The meta models, such as the resource meta model, and the reflective architecture in general need to be modified to fit the new configurations. (ii) A cloud orchestrator and a unified, modular, reusable deployment topology and orchestration specification like TOSCA are essential components of the current architecture. There is a need for such an abstract specification that allows interaction with the state-of-the-art container orchestration systems (e.g., Kubernetes, Mesos, Docker Swarm, etc.) because mapping TOSCA to containers is not fully realized [60]. (iii) Because engineering workflows include a wide range of domain tools that can be executed on different operating systems, those operating systems should fully support containers. (iv) Finally, in this situation, a novel auto scaling technique ought to be suggested.

Dynamic Adaptation Policies A policy-based approach to defining rules, and specifically their thresholds, is the foundation of the present InfraComposer architecture (see Section 3.4.3, 3.5). In order to prepare the policies, this method mainly depends on the domain knowledge of engineering workflows. In addition, thresholds are statically defined. Chen et al. [49] divide the state of the art into two categories in order to dynamicize the decision-making mechanism: (i) control theoretic techniques, and (ii) search-based optimization. Though they have been thoroughly explored, control theoretic techniques (e.g., Kalman control, fuzzy control, or PD) are not very good at handling multi-objectivity (e.g., execution time, cost, etc.) or performing well in situations when a lot of rescaling decisions need to be made.

Threats to validity In an Open Stack cloud platform that is shared with multiple tenants, we assessed the middleware. To ensure that co-located tenants on the bare-metal compute nodes do not affect our performance metrics, we have considered every isolation method that is feasible. Additionally, industry-standard tools and workflows, as well as actual aircraft designs, serve as the foundation for the tested application cases. We were only able to execute our evaluations for a brief period of time because these instruments and data were quite expensive.

3. Conclusions

To achieve intelligent and optimal deployment on cloud infrastructures, we presented InfraComposer, a policy-driven, adaptive, and reflective middleware that facilitates simulation and optimization workflows. Our methodical process consists of: (i) getting the engineers' feedback on the first direct workflow deployment using annotations, and (ii) learning new things from the real execution history used to create better deployments. Policies use execution histories to optimize deployments by encapsulating the expertise of system administrators and cloud engineers. Reconfiguring and expanding the execution environment for subsequent engineering workflow iterations is one way that these policies respond to reflective time-series data. We demonstrated particular adaptive deployment scenarios in actual application cases within the aeronautics area as a means of validating and assessing the middleware. We verified that the middleware's reflective and adaptable features could handle every situation.

References

[1] Acar, A., Aksu, H., Uluagac, A. S., and Conti, M. A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys (CSUR) 51, 4 (2018), 1–35.
[2] Acar, Y., Backes, M., Fahl, S., Garfinkel, S., Kim, D., Mazurek, M. L., and Stransky, C. Comparing the usability of cryptographic apis. In 2017 IEEE Symposium on Security and Privacy (SP) (2017), IEEE, pp. 154–171.
[3] Agache, A., Brooker, M., Iordache, A., Liguori, A., Neugebauer, R., Piwonka, P., and Popa, D.-M. Firecracker: Lightweight virtualization for serverless applications. In 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20) (Santa Clara, CA, Feb. 2020), USENIX Association, pp. 419–434.
[4] Agarwal, A., and Kamara, S. Encrypted distributed hash tables. IACR Cryptol. ePrint Arch. 2019 (2019), 1126.
[5] Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y. Order preserving encryption for numeric data. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data (2004), ACM, pp. 563–574.
[6] Al-Dhuraibi, e. a. Elasticity in cloud computing: state of the art and research challenges. IEEE Transactions on Services Computing (2017).
[7] Alashwali, E. S., and Rasmussen, K. What's in a downgrade? a taxonomy of downgrade attacks in the tls protocol and application protocols using tls. In International Conference on Security and Privacy in Communication Systems (2018), Springer, pp. 468–487.
[8] Amann, S., Nadi, S., Nguyen, H. A., Nguyen, T. N., and Mezini, M. Mubench: A benchmark for api-misuse detectors. In Proceedings of
[9] Arciszewski, S. Against cipher agility in cryptography protocols, 2019. https://paragonie.com/blog/2019/10/against-agility-incryptography- protocols (Last visit: 2021-06-17).
[10] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., et al. Serverless computing: Current trends and open problems. In Research Advances in Cloud Computing. Springer, 2017, pp. 1–20.
[11] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., and Suter, P. Serverless Computing: Current Trends and Open Problems. Springer Singapore, 2017, pp. 1–20.
[12] Barker, E., and Dang, Q. NIST special publication 800-57 part 1, revision 4. NIST, Tech. Rep 16 (2016).

[13] Bellare, M., Boldyreva, A., and O'Neill, A. Deterministic and efficiently searchable encryption. Cryptology ePrint Archive, Report 2006/186, 2006. https://eprint.iacr.org/2006/186.
[14] Beni, E. H., Lagaisse, B., and Joosen, W. WF-interop: adaptive and reflective rest interfaces for interoperability between workflow engines. In Proceedings of the 14th International Workshop on Adaptive and Reflective Middleware (2015), ACM, p. 1.
[15] Beni, E. H., Lagaisse, B., and Joosen, W. Adaptive and reflective middleware for the cloudification of simulation & optimization workflows. In Proceedings of the 16th Workshop on Adaptive and Reflective Middleware (2017), pp. 1–6.
[16] Beni, E. H., Lagaisse, B., and Joosen, W. Infracomposer: Policydriven adaptive and reflective middleware for the cloudification of simulation & optimization workflows. Journal of Systems Architecture 95 (2019), 36–46.
[17] Beni, E. H., Lagaisse, B., Joosen, W., Aly, A., and Brackx, M. Datablinder: A distributed data protection middleware supporting search and computation on encrypted data. In Proceedings of the 20th International Middleware Conference Industrial Track (2019), pp. 50–57.
[18] Beni, E. H., Truyen, E., Lagaisse, B., Joosen, W., and Dieltjens, J. Reducing cold starts during elastic scaling of containers in kubernetes.
[19] Bernstein, D. Nacl, 2008. https://nacl.cr.yp.to/ (Last visit: 2021-06-17).
[20] Bernstein, D. Containers and Cloud: From LXC to Docker to Kubernetes. IEEE Cloud Computing 1, 3 (2014), 81–84.
[21] Biryukov, A., Velichkov, V., and Corre, Y. L. Automatic search for the best trails in arx: Application to block cipher Speck. Cryptology ePrint Archive, Report 2016/409, 2016. https://eprint.iacr.org/2016/409.
[22] Blair, G., Costa, F., Coulson, G., Delpiano, F., Duran, H., Dumant, B., Horn, F., Parlavantzas, N., and Stefani, J.-B. The design of a resource-aware reflective middleware architecture. In Meta-Level Architectures and Reflection (1999), Springer, pp. 115–134.
[23] Blair, G. S., Coulson, G., Robin, P., and Papathomas, M. An architecture for next generation middleware. In Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (2009), Springer-Verlag, pp. 191–206.
[24] Blochberger, M., Petersen, T., and Federrath, H. Mitigating cryptographic mistakes by design. Mensch und Computer 2019- Workshopband (2019).
[25] Boldyreva, A., Chenette, N., Lee, Y., and O'Neill, A. Orderpreserving symmetric encryption. Cryptology ePrint Archive, Report 2012/624, 2012. https://eprint.iacr.org/2012/624.
[26] Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., and Zimmerman, J. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation, 2015.
[27] Bösch, C., Hartel, P., Jonker, W., and Peter, A. A survey of provably secure searchable encryption. ACM Computing Surveys (CSUR) 47, 2 (2014), 1–51.
[28] Bösch, C., Hartel, P., Jonker, W., and Peter, A. A survey of provably secure searchable encryption. ACM Computing Surveys (CSUR) 47, 2 (2015), 18.
[29] Bost, R. Sophos: Forward secure searchable encryption. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (2016),
[30] Bost, R., Minaud, B., and Ohrimenko, O. Forward and backward private searchable encryption from constrained cryptographic primitives. Cryptology ePrint Archive, Report 2017/805, 2017. https://eprint.iacr.org/2017/805.
[31] Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) 6, 3 (2014), 1–36.
[32] Breitenb ucher, U., Binz, T., Képes, K., Kopp, O., Leymann, F., and Wettinger, J. Combining declarative and imperative cloud application provisioning based on TOSCA. In 2014 IEEE International Conference on Cloud Engineering (2014), pp. 87–96.
[33] Brewer, E. A. Kubernetes and the path to cloud native. In Proceedings of the sixth ACM symposium on cloud computing (2015), pp. 167–167.
[34] Browne, P. JBoss Drools business rules. Packt Publishing Ltd, 2009.
[35] Bulck, J. V., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., Silberstein, M., Wenisch, T. F., Yarom, Y., and Strackx, R. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In 27th USENIX Security Symposium (USENIX Security 18) (Baltimore, MD, Aug. 2018), USENIX Association, p. 991–1008.
[36] Burns, J., Moore, D., Ray, K., Speers, R., and Vohaska, B. Ec-oprf: Oblivious pseudorandom functions using elliptic curves. IACR Cryptology.

Author Profile

Dr. Dilshad Ghani received the Ph.D Magadh University Bodh-Gaya Bihar India in 2024. Certified MCSE, CCNA, ITILV3, Microsoft Certified Azure Administrator Associate Certified and Certified AWS Solutions Architect with more than 15 years of IT experience in IT infrastructure and Networking. Expertise in Network, Windows, Cloud computing and Linux systems.