Experimental Validation of the Suitability of Virtualization-Based Replication for Fault Tolerance in Real-Time Control of Electric Grids

Seyed Alireza Sanaee Kohroudi, Jalal Mostafa, Maaz Mohiuddin, Wajeb Saab, Jean-Yves Le Boudec
IC-LCA2 EPFL, Switzerland
{firstname.lastname}@epfl.ch

ABSTRACT

Real-time control systems (RTCSs) perform complex control and require low response times. They typically use third-party software libraries and are deployed on generic hardware, which suffer from delay faults that can cause serious damage. To improve availability and latency, the controllers in RTCSs are replicated on physical nodes. As physical replication is expensive, we study the alternative of exploiting virtualization technology to run multiple virtual replicas on the same physical node. As virtual replicas share the same resources, the delay faults they experience might be correlated, which would make such a replication method unsuitable. We conduct several experiments with an RTCS for electric grids, with multiple virtual replicas of its controller. We find that although the delay of a virtual machine is higher than of a physical machine, the correlation between high delays among the virtual replicas is insignificant, causing an overall improved availability. We conclude that virtual replication is indeed applicable to certain RTCSs, as it can improve reliability without added cost.

CCS CONCEPTS

• **Computer systems organization → Availability**; • **Software and its engineering → Real-time systems software**;

KEYWORDS

Measurements, Real-Time Control Systems, High Availability, Virtualization, Replication, Delay Faults

ACM Reference Format:
Seyed Alireza Sanaee Kohroudi, Jalal Mostafa, Maaz Mohiuddin, Wajeb Saab, Jean-Yves Le Boudec. 2018. Experimental Validation of the Suitability of Virtualization-Based Replication for Fault Tolerance in Real-Time Control of Electric Grids. In *ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '18), October 11–12, 2018, Oulu, Finland.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3239235.3267429

1 INTRODUCTION

Real-time control systems (RTCSs) are a subset of cyber-physical systems that have a response time in the range of a few hundred milliseconds. Prominent examples are systems that monitor and control electric grids [2, 8]. These systems are mission-critical as a failure can have severe economic and safety consequences [1, 6].

Traditionally, RTCSs are deployed using embedded or programmable logic controllers (PLCs) [9]. However, the increasing complexity of the control frameworks has led to the use of commercial off-the-shelf (COTS) industrial computers. Industrial computers use standard operating systems; and the control software uses off-the-shelf software. In contrast to PLCs that are known for hard real-time performance, the industrial computers are soft real-time and might miss the required target response times. In other words, the controllers designed with COTS components are particularly susceptible to delay faults [10, 11, 13], besides crash faults. In order to mask these faults, RTCSs replicate their controllers on physical machines (referred to as physical replicas), so that if one controller experiences a fault, another replica will be available.

Physical controller replication increases the cost of deployment and maintenance. Advances in virtualization technology enable virtual machines (VMs) to provide near-native performance [12]. Such performance, coupled with the low cost of VMs, has led to the use of virtualization-based replication schemes, whereby multiple virtual-machine replicas (hereafter referred to as virtual replicas) are deployed on the same physical node [3, 5, 7]. This approach has been applied to both benign and Byzantine fault-tolerance. However, it has not been applied to RTCSs with sub-second response time, where delay-faults are of importance.

Before using virtualization for replication in RTCSs, two questions need to be answered: (1) As each virtual replica has less computing, memory, and network resources, how does the response time of the virtual replicas compare to the response time of a physical replica? (2) As all virtual replicas share underlying hardware, host operating system, and hypervisor, how are the delay faults on virtual replicas correlated? By answering these questions, we can quantify the reliability of virtualization-based replication mechanisms and compare it to that of physical replication schemes.

For this evaluation, we have chosen COMMELEC [2], an RTCS for electric grids. We perform several experiments with the COMMELEC controller running on (1) a single physical machine, and (2) multiple VMs on the same physical machine. We use the same code of the COMMELEC controller, as used in a real deployment in a CIGRÉ benchmark low-voltage microgrid [14]. We find that the delays of a VM are higher than that of a physical machine. However, the lack of a significant correlation between the instances

Supported by the "SNSF - NRP 70" Energy Turnaround project and SCCER-FURIES.

ESEM '18, October 11–12, 2018, Oulu, Finland
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5823-1/18/10…$15.00
https://doi.org/10.1145/3239235.3267429

of the virtual replicas results in the three virtual replicas having availability higher than the one physical replica.

2 RELATED WORK

To the best of our knowledge, this is the first work on evaluating the real-time performance of an RTCS under virtualization. Our work is inspired by [15], where the authors compare the performance of web applications running on a physical machine to that of the same applications running on VMs. In contrast to RTCSs, web applications are not as mission critical and do not have stringent real-time requirements. So, the authors in [15] focus their attention on the throughput of the VMs and side-line the response time. Also, the availability of a web server can be easily measured by the number of client requests served. However, the availability of RTCSs is not trivial to formulate and depends on the underlying physical process, as discussed in [10].

Recent studies demonstrate that the performance of VMs is approaching near-native performance [12]. This trend of virtualization, coupled with its lower cost when compared to investing in physical replicas, has made it an obvious choice for redundancy in fault-tolerance architectures [3, 5, 7]. These works propose algorithms for managing VMs in the presence of crash faults [5] or Byzantine faults [3, 7], to minimize overhead in terms of delay due to replication. The performance of these fault-tolerance architectures depends heavily on the presence of enough virtual replicas available to take part in the fault-tolerance algorithm. In this paper, we focus on quantifying the availability of the virtual replicas, i.e., the presence of non-delayed, non-crashed replicas. Thus, the results obtained here shed light on the feasibility of the fault-tolerance architectures [3, 5, 7] in RTCSs.

The trend to use COTS components [13] in RTCSs, along with the existence of fault-tolerance architectures such as [3, 5, 7], has led to efforts in the development of real-time hypervisors [4, 16]. However, these projects are still in early development phases. Thus, we limit our performance evaluation to the established Xen hypervisor.

3 THE COMMELEC FRAMEWORK

For our measurement campaign, we used the COMMELEC framework [2], as an RTCS for electric grids. In this RTCS, there is a central controller, referred to as the grid agent (GA); it communicates with one or more resource agents (RAs), each responsible for an individual electric resource such as a battery or a photo-voltaic panel. The control is done in rounds of 60-100 ms. In each round, the GA receives one measurement from each RA, computes and issues setpoints of the RAs. Each RA, upon reception of a setpoint, implements it and waits to accurately sense the change of state (60 ms) before responding with new measurements for the next round.

The goal of the GA is to keep the electric grid in a feasible state, i.e, respect the ampacity limits of the lines in the grid and prevent over- and under-voltages. Consequently, the GA must maintain continuous control of the grid to avoid undesirable effects. So, the authors of [2] propose to control the electric grid at least once every 100 ms. Failure of the GA to remain *available* might result in violations of the grid constraints, potentially causing a blackout.

An equally important requriement in control is not to implement old and outdated setpoints, as they could result in a control behavior that was unaccounted for by the GA. Hence, as described in [10], we say a setpoint is unsafe to implement if the duration between the implementation and the reception of the measurements used for its computation is larger than a certain threshold (τ_o). With a cycle time of 100 ms, of which 60 ms is spent waiting for the duration of RA sensing, gives an upper-bound of 40 ms on τ_o as. When the setpoint is unsafe, we say the GA is delay faulty.

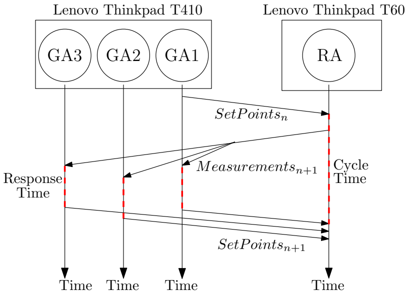


Figure 1: Experimental setup with three virtual GA replicas

4 MEASUREMENT SETUP

The goal of our measurement campaign is to ascertain if virtualization-based replication is suitable for mission-critical RTCSs such as COMMELEC. For this, we perform two sets of experiments with a GA and an RA. In the first set, we run the GA executable on a laptop (with same specs as an industrial PC). In the second set, we deploy the Xen hypervisor on the laptop computer, and instantiate three VMs, each running the same GA executable, as shown in Fig. 1. The first setup emulates the current deployment of COMMELEC [14], and the second setup emulates a virtualization-based fault-tolerance as discussed in the literature [3, 5, 7]. In order to tolerate f faults, the minimum number of replicas needed is $2f + 1$. So, we use three replicas, need to tolerate 1 faulty replica.

For the GA, we use Lenovo T410, with Debian 9 and kernel v4.9.3. It has an Intel Quad Core i7 processor and 4GB DRAM. In the experiments with the Xen hypervisor, each VM is provided with 1 GB DDR3 RAM and one core that is pinned to an isolated physical core. Note that, with respect to the Xen architectural style, the hypervisor is acting as a bridge for I/O operations. This isolation of the hypervisor's CPU ensures that the I/O operations are performed without interference from the co-located VMs.

In each COMMELEC cycle, we record the response time of the GAs, as the time between reception of measurements and issuing of setpoints at each GA replica. We also record the duration of a COMMELEC cycle as the time between two consecutive receptions of measurements at an RA (see Fig. 1). This approach to measuring cycle time is equivalent to the time between two consecutive setpoint receptions, but is easier to measure in the case of replicated GAs that send multiple setpoints to the RA.The response time of each GA is used to characterize the rate of unsafe setpoints, and the cycle time is used to characterize whether the GA is able to keep up with the 100 ms target required by COMMELEC. In the scenario with virtual replicas, due to varying I/O wait-times, the 3 replicas might start computation at different times. To time-align the COMMELEC cycles on the 3 replicas, we use the Precision Time Protocol for time-synchronization, with an accuracy of < 1 ms.

5 EXPERIMENTAL METHODOLOGY

We present results from 5 million samples from each controller: for scenario 1 with the GA running on a physical machine, and scenario 2 with the GA replicas running on VMs on the same physical machine. This is around 5.5 days of COMMELEC runtime. We use the response time of the GAs to compute two metrics: delay-fault rate (DFR) and availability.

DFR is defined as the fraction of COMMELEC cycles during which the GA is delay faulty. A replica is considered delay faulty if the response time is larger than τ_o. We obtain DFR as a function of τ_o by varying τ_o in the interval [10, 40] ms. $\tau_o = 10$ ms represents high-end scenarios such as unintentional island (disconnection from the main grid and reconnection, and $\tau_o = 40$ ms represents low-end scenarios such as following an external dispatch plan.

While DFR gives an insight into the frequency of occurrence of faults, it does not capture the effect of duration of delay faults. For example, if a delay fault in a GA is larger than the maximum cycle time (100 ms), then it affects at least two consecutive COMMELEC cycles. Thus, a joint-DFR for multiple GA replicas does not have any significance. Hence, we use another metric for evaluation, namely availability, first defined for delay-faulty RTCSs in [10].

We say the GA is available, at a time instant t, if the RA has implemented a setpoint in the interval $[t - 100\ ms, t]$; thereby respecting COMMELEC framework’s control deadline. Consequently, we compute availability using a sliding window across the entire runtime of the experiment, and by taking the fraction of 100 ms intervals during which the GA issued safe setpoints.

The DFR of a GA running on a physical machine is expected to be lower than that of a GA running on a VM because of lower memory and CPU resources on a VM, and contention between VMs for the shared resources. However, as the GAs on the VMs are replicated, their combination might have higher availability than the controller running on a physical machine.

6 RESULTS

6.1 Response and Cycle Times

Table 1 shows the mean, variance, 99%-ile, 99.99%-ile and the maximum response and cycle times of the GA running on a physical machine (Physical) and virtual replicas (VM_i). We notice that the GAs running on VMs have similar response times. The mean response time of the GA running on the physical machine is 5-8 times smaller than that of the GAs running on the VMs. This difference is more pronounced in the tail response-time with the 99%-ile of the virtual replicas being 10-14 times that of the physical machine. As noted earlier, this behavior is expected as VMs have shared resources and face contention for the physical resources.

Although there is a large deviation in the tail response times from the mean response time of both the physical machine and the VMs, the corresponding impact on the cycle time is low. This is because when the response time is low (below 10 ms), majority of the COMMELEC cycle is spent in sensing the state of the grid accurately so as to improve the efficiency of the control. However, when the response time is large (about 18 ms at 99.99%-ile), there is a substantial impact of delay faults on the cycle time as observed by high values of cycle time at the same quantile.

It is astonishing to see that a single VM can leave the grid uncontrolled for up to a second, i.e., 10x of the required nominal control cycle of 100 ms. Such behaviour is highly unsafe for real-time operation of the grid and can result in catastrophes [1]. We conclude that although single VMs can, on average, maintain the desired cycle time, they are far from useable, due to a heavy-tailed delay distribution.

Metric [ms]	Physical	VM_1	VM_2	VM_3
Mean	0.47, 61.04	2.54, 63.14	3.78, 63.12	3.78, 63.12
Variance	0.47, 0.47	3.82, 4.47	3.92, 4.78	3.82, 4.66
99%-ile	1.09, 61.68	11.56, 72.71	15.28, 75.29	15.33, 75.31
99.99%-ile	1.89, 62.52	16.69, 79.94	18.46, 80.91	18.44, 80.30
Max	305.67, 366.162	1006.46, 1067.63	1048.98, 1109.25	997.31, 1057.46

Table 1: Comparison of response and cycle times. 1^{st} entry in each column is response time and 2^{nd} entry is cycle time.

6.2 DFR and Correlation of Response Times

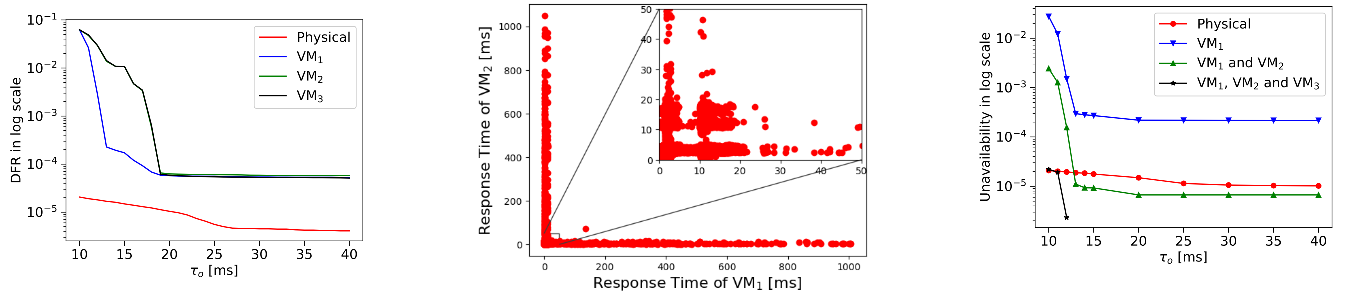
Next, we look at the DFR of the physical GA, and for each virtual replica. As explained in §5, we vary τ_o in the range $[10, 40]$ ms. Fig. 2a shows the DFR as a function of τ_o for the GAs running on the physical machine and the three VMs. We find that for time-sensitive applications such as islanding and reconnection that have $\tau_o = 10$ ms, each single VM has a DFR of $6.2E - 2$, whereas a physical machine for the same threshold has a DFR of $2.1E - 5$. However, at the other end of the spectrum, for $\tau_o = 40$ ms, the DFR of a single VM is $5.3E - 5$ and that of the physical machine is $4.1E - 6$. Thus, we find that while a single VM is certainly not suitable for time-sensitive operations, but its DFR is comparable to that of the physical machine for less time-sensitive operations.

To get a closer look at how the delay faults in the VMs are related, we look at the correlation of delays between them. Fig. 2b show the pairwise scatter plots of delays of VM_1 and VM_2. The scatter plots for VM_2, VM_3, and VM_1, VM_3 are similar. At the outset, the absence of points along the line $y = x$ shows that there is no correlation between the delays larger than 100 ms. However, zooming in at the range of delays below 50 ms, we see that delays around 10 ms are highly correlated. This indicates that the combination of the VMs might not be highly available for time-sensitive applications with low τ_o due to simultaneous delay faults in multiple virtual replicas. We explore this further with the availability metric.

6.3 Availability

Finally, we measure the availability in each experiment, where availability is defined as the number of 100 ms intervals in which the RA received a valid setpoint. Valid setpoints are ones with response time smaller than the threshold τ_o. We compare the availability of the physical machine with that of one, two, and three virtual replicas, over several values of τ_o. The results are shown in Fig. 2c. We observe that the unavailability of one virtual replica is the highest (results are similar for VMs other than VM_1), and that it decreases with the addition of another virtual replica (results are also similar with different VM pairs).

Compared to the unavailability of the physical machine, the unavailability of two virtual replicas is higher for smaller values



(a) DFR of physical and virtual replicas varying with τ_o.

(b) Pairwise scatter plots of response times of virtual replicas VM$_1$ and VM$_2$.

(c) Unavailability as a function of τ_o. For $\tau_o > 11$, no unavailability in 5 million runs with 3 virtual replicas

Figure 2: Results on DFR, scatter plot of response times, and unavailability

of the threshold ($\tau_o < 13$), but it becomes lower for larger values. For three virtual replicas, the unavailability is comparable to that of the physical machine for $\tau_o < 11$. However, for larger τ_o, we found no unavailability in 5 million runs, thus, concluding that the availability at 95% confidence is in the range $(0, 7E-7]$.

This result is to be expected from the lack of correlation observed in Fig. 2b of the threshold higher than 11. The 3 VMs never experience, all at the same time, a delay fault resulting in a response time greater than 11 ms. This results in availability of 1 in those cases.

It is clear from Fig. 2c that using three replicated VMs significantly improves the availability of the GA over one physical machine, for $\tau_o \geq 13$ ms. To further confirm this result, we performed a statistical test for all values of τ_o in the interval $[10, 40]$ ms. We split the data into 20 sub-experiments each representing approximately 6 hours of execution time with 250,000 samples. We used these sub-experiments to obtain two series of availabilities, one each for the physical GA and replicated virtual GA, for each value of τ_o. Then, we applied the two-sided student t-test, to obtain one p-value for each value of τ_o. For all values of $\tau_o \geq 12$ ms, the value was close to 0, indicating a strong significance in availability improvement. For $\tau_o = 10$ ms and $\tau_o = 11$ ms, however, we obtained p-values of 0.95 and 0.65, respectively, which does not indicate any significant improvement in availability due to the replicated virtual GAs. As three virtual replicas were enough to improve the availability over one physical machine, we did not experiment with additional replicas, also due to the limited number of cores.

7 CONCLUSION

We study, via experimental validation, if virtualization-based replication is a suitable method for improving reliability in real-time control of electric grids. We perform two sets of experiments using the actual code of an RTCS of electric grids: COMMELEC. The first set involves running the COMMELEC code on a physical machine, and the second set involves running three replicas of the code, each on a VM, co-located on the same physical machine. We observe that each VM has higher response times, subsequently a higher DFR, than the physical machine. Whereas, we observe that the correlation of delay faults is insignificant between the virtual replicas. This results in an overall improvement in availability at no extra hardware cost. Consequently, exploiting virtualization is indeed a suitable replication method for RTCS. We plan to deploy the virtualized architecture of COMMELEC in the on-campus microgrid, and to obtain in-field measurements for a longer duration.

REFERENCES

[1] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, et al. 2005. Causes of the 2003 Major Grid Blackouts in North America and Europe and Recommended Means to Improve System Dynamic Performance . *Power Systems, IEEE Transactions on* 20, 4 (Nov 2005), 1922–1928. https://doi.org/10.1109/TPWRS.2005.857942

[2] Andrey Bernstein, Lorenzo Reyes-Chamorro, Jean-Yves Le Boudec, and Mario Paolone. 2015. A Composable Method for Real-Time Control of Active Distribution Networks with Explicit Power Setpoints. Part I: Framework. *Electric Power Systems Research* 125 (2015), 254–264.

[3] Byung-Gon Chun, Petros Maniatis, and Scott Shenker. 2008. Diverse Replication for Single-Machine Byzantine-Fault Tolerance.. In *USENIX Annual Technical Conference*. 287–292.

[4] Jiun-Hung Ding, Chang-Jung Lin, Ping-Hao Chang, Chieh-Hao Tsang, Wei-Chung Hsu, and Yeh-Ching Chung. 2012. ARMvisor: System Virtualization for ARM. In *Proceedings of the Ottawa Linux Symposium (OLS)*. Citeseer, 93–107.

[5] Tobias Distler, Ivan Popov, Wolfgang Schröder-Preikschat, Hans P Reiser, and Rüdiger Kapitza. 2011. SPARE: Replicas on Hold.. In *NDSS*.

[6] Michael Dunn. 2013. Toyota's Killer Firmware: Bad Design and its Consequences. *EDN Network, October* 28 (2013).

[7] Eric Keller, Minlan Yu, Matthew Caesar, and Jennifer Rexford. 2009. Virtually Eliminating Router Bugs. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 13–24.

[8] Junhao Lin, Ka-Cheong Leung, and Victor OK Li. 2014. Optimal Scheduling with Vehicle-to-Grid Regulation Service. *IEEE Internet of Things Journal* 1, 6 (2014), 556–569.

[9] G. Macher, M. Bachinger, and M. Stolz. 2017. Embedded Multi-core System for Design of Next Generation Powertrain Control Units. In *2017 13th European Dependable Computing Conference (EDCC)*. 66–72. https://doi.org/10.1109/EDCC.2017.32

[10] Maaz Mohiuddin, Wajeb Saab, Simon Bliudze, and Jean-Yves Le Boudec. 2016. Axo: Masking Delay Faults in Real-Time Control Systems. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*. IEEE, 4933–4940.

[11] Maaz Mohiuddin, Wajeb Saab, Simon Bliudze, and Jean-Yves Le Boudec. 2018. Axo: Detection and Recovery for Delay and Crash Faults in Real-Time Control Systems. *IEEE Transactions on Industrial Informatics* 14, 7 (2018), 3065–3075.

[12] M. Raho, A. Spyridakis, M. Paolino, and D. Raho. 2015. KVM, Xen and Docker: A Performance Analysis for ARM Based NFV and Cloud Computing. In *2015 IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*. 1–8. https://doi.org/10.1109/AIEEE.2015.7367280

[13] Donald J Reifer, Victor R Basili, Barry W Boehm, and Betsy Clark. 2004. Cots-Based Systems–Twelve Lessons Learned about Maintenance. In *COTS-Based Software Systems*. Springer, 137–145.

[14] Lorenzo Reyes-Chamorro, Andrey Bernstein, Niek J Bouman, Enrica Scolari, Andreas Kettner, Benoit Cathiard, Jean-Yves Le Boudec, and Mario Paolone. 2018. Experimental Validation of an Explicit Power-Flow Primary Control in Microgrids. *IEEE Transactions on Industrial Informatics* (2018).

[15] Luis Moura Silva, Javier Alonso, and Jordi Torres. 2009. Using Virtualization to Improve Software Rejuvenation. *IEEE Trans. Comput.* 58, 11 (2009), 1525–1538.

[16] Sisu Xi, Justin Wilson, Chenyang Lu, and Christopher Gill. 2011. Rt-Xen: Towards Real-Time Hypervisor Scheduling in Xen. In *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*. IEEE, 39–48.