Computer Standards & Interfaces 49 (2017) 11–21

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Analyzing point-to-point DDS communication over desktop virtualization software

Marisol García-Valls *, Pablo Basanta-Val

Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid, Av. de la universidad 30, 28911 Leganés, Madrid, Spain

ARTICLE INFO

Article history:
Received 29 February 2016
Received in revised form 23 June 2016
Accepted 30 June 2016
Available online 8 July 2016

Keywords:
Middleware
DDS
Virtualization software
VirtualBox
Performance

ABSTRACT

Virtualization technologies introduce additional uncertainty and overhead for distributed applications, that may challenge their timeliness. The additional software abstraction layers of the virtualization software offer powerful parallel execution environments but, at the same time, reduce the effective performance as additional software layers are introduced. This requires some fine tuning of, among others, the communication middleware software. The different resource management mechanisms of the middleware may collide with the specific algorithms replicated by the underlying virtualization software. The present work characterises the set of steps of a publish-subscribe communication middleware in a distributed system, enhancing it to suit the communication scheme of virtualized remote nodes. The potential communication steps are identified, and the overhead introduced by the execution over the virtualization software is provided by means of a data communication rate metric; a set of benchmark tests are presented that empirically evaluate the overhead and stability of the most widely used publish-subscribe (P/S) middleware named DDS (Data Distribution System for real-time systems). A general purpose cloud computing virtual machine monitor is utilized to identify the side effects and the drawbacks of the general virtualization technology. Results are obtained on a setting with two different DDS implementations over a general purpose virtualization software, together with a discussion about the potential drawbacks of the main communication operations.

1. Introduction

Communication middleware and virtualization technologies are two main contributions to the development of distributed systems, easing the maintenance of large systems and enhancing their computation capacity [46,48]. On the one hand, middleware allows abstracting the low level details of the networking protocols and of the physical platforms (e.g. endianness, frame structure, and packaging, among others); this augments the productivity of systems development by easing the programmability and debugging. More recently, virtualization technologies have promoted a new technological trend that has fastly penetrated different domains due to the benefits that it brings about: a) speed up of the customized system development and deployment to specific platforms; b) server consolidation and the subsequent savings on energy; c) reduction of maintenance and deployment costs, and d) data availability any time anywhere.

Communication middleware and virtualization technology originated for general purpose distributed applications, so initially in a different perspective from that of real-time environments where the response time bounds of the system operation need to be found. As science evolves and new applications are envisioned and engineered, real-time applications have progressively approached middleware and virtualization technologies, facing the problem of temporal predictability [5]. However, achieving time-deterministic behavior in the presence of communication middleware is still an area of research that is even more challenging over a virtualized platform. As virtualization technology continues to flood the market, the middleware connecting application-level logic will run both over virtualization software and also over bare machines; such heterogeneous settings have the potential to alter the performance experienced by the middleware and, subsequently, by applications. These performance variations need to be analysed in time-sensitive domains such as soft real-time applications.

For real-time applications, research has primarily focused on developing virtualization technology that ensures time-deterministic execution; this is typically done by fine tuning the real-time knobs of the specific used technology such as KVM [2] or Xen [3]. Nevertheless, these works are silent about the behavior of the distributed applications and the performance of the communication middleware that can execute on top of them. The negative and unpredictable consequences of considering the networked communications and middleware in the

* Corresponding author.
E-mail addresses: mvalls@it.uc3m.es (M. García-Valls), pbasanta@it.uc3m.es (P. Basanta-Val).

http://dx.doi.org/10.1016/j.csi.2016.06.007
0920-5489/

distributed applications is not typically addressed in this domain, as real-time research typically makes heavy assumptions with respect to the behavior of the network, and the communication middleware that is used is mainly neglected.

For networked real-time systems, the main focus has been on eliminating (or minimizing) the sources of unpredictability by direct programming of tasks in the real-time operating system or directly in the hardware platform itself and using controlled medium access protocols to develop real-time networks. However, middleware has typically been implemented for distributed systems over non collision-free networks, and using software engineering techniques that introduce additional software layers aiming at easing programmability and interoperability. As a consequence, communications middleware has appeared as a black box, containing extra code that is difficult to analyse with sufficient level of detail and guarantees as generally required by real-time applications.

Distributed real-time applications have incorporated the usage of middleware since the times of Corba and, precisely, RTCorba. However, the complexity of that standard and the lack of competing implementations that would support the complete standard was not a favorable context to increase the popularity of middleware for real-time domains. The appearance of the OMG's DDS standard [1] (Data Distribution Service for Real-Time Systems) that provides asynchronous interoperability via a data-centric publish-subscribe paradigm has promoted the usage of communication middleware for *distributed soft real-time applications* again. It has become the de facto standard for some application domains such as DoD projects that mostly have time requirements or directly fall in the category of real-time systems. One of the success factors of DDS is that it defines a collection of diverse QoS parameters that promise to fine-tune the behavior of the communications. In general, the level of temporal guarantees provided by different DDS implementations varies depending on different factors such as the underlying physical deployment, network, and the application code. There are not many published independent studies about the performance achieved by the different implementations, and (up to the best of our knowledge) there are no studies that analyze its communication structure in virtualised environments in order to detect the design bottlenecks of the middleware in relation to cloud computing technology.

There are different virtualization techniques such as those that use software containers for operating systems such as Linux (e.g. Docker [44] or Vagrant [45]) with the goal of avoiding the overhead of starting and running virtual machines. This technique supports the execution of independent containers in a sole Linux instance, providing an additional abstraction layer at system level and using techniques to achieve isolation of the kernel resources such as *cgroups* or *namespaces*.

In this paper, we analyze the behavior of the integration of DDS middleware over a general purpose virtualization software based on virtual machines and not on software containers to identify the potential drawbacks of such an integration with respect to the main operations involved in the communication. We identify the phases that need be executed on a bare communication among two participants, characterising their temporal cost and deriving a metric that indicates the data communication rate by different DDS implementations. It is not the objective of this contribution to fine tune the used VMM for real-time applications, but to serve as a baseline to explore the deficiencies of the communication structure; determining and fine tuning the temporal behavior of DDS over a real-time hypervisor is a natural follow up the presented work.

1.1. Problem description

The execution overheads of communication middleware over a general purpose virtualized infrastructure are the following: (1) the additional virtualization software layers; (2) overhead of the communication middleware abstractions; and (3) implementation optimization issues.

The sole execution of a VM in isolation experiences the overhead of the *virtualization layers* for different reasons such as the translation of the physical interrupts and their forwarding as software interrupts to the above application layers, depending on the virtualization strategy used. Also, when different VMs are in execution, the experienced overhead is enhanced by the natural competition for the machine resources. This may affect the use of *visible* shared resources (e.g. the same physical core or memory capacity) and *invisible* shared resources (e.g., cache space, memory bandwidth, etc.), being visible or invisible depending on the implementation of the host operating system and virtualization monitor.

The *overhead of the communication middleware abstractions* appears as the data messages have to traverse the software layers and are queued at different levels. This overhead is reflected in the main statistical metrics (i.e. minimum, average, and maximum response times), increasing jitter and overhead. The cost of marshalling and unmarshalling data in the communication is one of the major penalties, and part of it may be alleviated using virtual machines that run similar virtualized operating systems and hardware infrastructures.

In such a scenario, *implementation optimisation* issues are key to obtain some performance benefits. However, this can be a very complex task depending on the particular used combination of middleware, guest OS, VMM, and host OS, as different inefficiencies may appear (e.g. unnecessary or/and inefficient copies among middleware buffers, guest OS, host OS, and VMM buffers).

In the future, virtualized servers will be key in real-time deployments, but still there is no sufficient confidence on the virtualization technology apart from some specific ARINC 653 compliant operating systems that provide isolation through partitions. Therefore, this study contributes to analyse and identify the key communication steps in a virtualized environment in order to progressively remote as much uncertainty as possible from the virtualized distributed communication schemes. It is not the focus of the work to experiment with the specific quality of service policies provided by DDS to optimize the networking level. Instead, it is our goal to assess the performance of a DDS enabled distributed application in a virtualized setting. This allows analysing the stability of the interaction between communicating nodes in a soft real-time domain and to reason about the suitability of deploying distributed applications based on P/S middleware in a virtualized machine.

1.2. Contribution

In a previous work [6], we have performed a black box exploratory performance evaluation on virtualized environments for middleware, without analysis of the main operations that take place in the communication. In this paper, we analyse the behavior resulting from the specific integration of complex and (in principle) independent COTS to set a starting point to analyze the suitability of the middleware architectures and communication process. Significant advances are that we analyze the internal structure of the middleware, providing a metric to identify the different performance drawbacks observed. Precisely we extend our previous work: (1) with a detailed description of the middleware structure comparing the deployment over bare machine with the one over a virtualised environment; (2) we elaborate on the *data communication frequency* calculating the communication saturation level in this setting, with performance indicators such as the *data communication rate* related to the cost of the virtualisation and in relation to the additional middleware operations; and (3) we provide a comparison of both execution types in bare machine and virtualised over general purpose VMMs, extending the set of tests to determine the invocation frequencies that are allowed.