An Open-Source Extendable, Highly-Accurate and Security Aware Simulator for Cloud Applications

Andreas Brokalakis[†], Nikolaos Tampouratzis[*], Antonios Nikitakis[†], Ioannis Papaefstathiou[†], Stamatis Andrianakis[*], Apostolos Dollas[*], Marco Paracchini[‡], Marco Marcon[‡], Danilo Pietro Pau[§], Emanuele Plebani[§]

[*]Telecommunications Systems Institute, Technical University of Crete, Greece
[†]Synelixis Solutions Ltd, Chalkida, Greece
[‡]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy
[§]Advanced System Technology, STMicroelectronics, Agrate Brianza, Italy

***Abstract*—In this demo, we present COSSIM, an open-source simulation framework for cloud applications. Our solution models the client and server computing devices as well as the network that comprise the overall system and thus provides cycle accurate results, realistic communications and power/energy consumption estimates based on the actual dynamic usage scenarios. The simulator provides the necessary hooks to security testing software and can be extended through an IEEE standardized interface to include additional tools, such as simulators of physical models. The application that will be used to demonstrate COSSIM is mobile visual search, where mobile nodes capture images, extract their compressed representation and dispatch a query to the cloud. A server compares the received query to a local database and sends back some of the corresponding results.**

I. Introduction

One of the main problems that designers of cloud and parallel applications face is the lack of simulation tools and models for system design and analysis. This is mainly because the majority of the existing simulation tools can handle efficiently only parts of a system (e.g. only the processing or only the network). Moreover, most of the existing simulators need extreme amounts of processing resources while faster approaches cannot provide the necessary precision and accuracy. COSSIM is an open-source framework that seamlessly simulates, in an integrated way, the networking and the processing parts of a system that executes a networked application. In addition, COSSIM supports accurate power estimations while it is the first tool supporting security as a feature of the design process.

The COSSIM simulation framework is built upon several established open-source packages. GEM5 [1] is used to simulate the processing components of each node in the system, while OMNET++ [2] is employed to simulate the real networking infrastructure between those nodes. To bind the whole framework together and establish a common notion of time, COSSIM employs the IEEE1516 HLA architecture through the open-source CERTI package. Security and robustness testing tools (based on custom, internally-developed Fuzz testing & DOS detection security components) are added through custom hooks to both GEM5 and OMNET++ simulators.

II. The COSSIM Framework in Detail

A. Processing Simulator

GEM5 [1] can simulate single or multi-core homogeneous or heterogeneous systems including their peripherals. The main engine of the simulator is ISA agnostic and it can support a broad range of ISAs. While GEM5 does not support natively any power/energy estimations, in COSSIM it is integrated with McPAT [3] to provide such information.

Besides the integration with McPAT, GEM5 has been modified to fit the COSSIM framework requirements. The most prominent changes are related to the network model of the simulator. More specifically, GEM5 supports networking through a virtual dummy link that emulates a cable over which Ethernet packets are sent and received without any delay (no switching or routing functionality is implemented). This conceptual cable link implies that GEM5 only supports simulation of two systems and a further restriction is that these two systems are handled in a single process and therefore they have to be identical (i.e. each system has the same processor and components).

In the scope of COSSIM, these limitations are unacceptably restrictive. Therefore Etherlink has been modified so that typical network protocols and topologies can be supported. COSSIM's approach is to tap the Ethernet packets from Etherlink and send them to a Networking Simulator modifying at the same time the packets to match the specific network protocol required by the application (i.e. Ethernet, WiFi, 3G, etc). To achieve this interconnection with the network simulator, CERTI HLA interfaces [4] have been employed. A custom library (*COSSIMlib*) has been integrated to the main core of GEM5 through Etherlink. This library acts as a wrapper to an RTI Ambassador Class serving to exchange messages over the network with the HLA Server (RTIG process) via TCP (and UDP) sockets. *COSSIMlib* exchanges Ethernet Packets captured from the Etherlink Device and sends (and accordingly receives) them to (from) the HLA Server (RTIG). The HLA Server forwards these messages to a proper interface of a Network Simulator that implements all the network related functionality (NIC physical layer and actual network).

By following this approach, COSSIM overcomes all network related limitations of GEM5 as there are no limitations for identically configured systems nor is there a restriction on the number of GEM5 systems that participate in the network. A very important additional benefit is that parallelism can be extracted at the process level, since multiple GEM5 instances can be run in parallel. On top of that, as CERTI HLA functions over IP, the different GEM5 instances do not even need to be on the same physical machine and the overall COSSIM simulator can thus be considered a distributed implementation.

The parallel nature of this implementation requires a mechanism to synchronize the different GEM5 instances and the network simulator. The need for synchronization arises from the non-fixed notion of time. GEM5 is an eventdriven simulator that schedules operations (events) on clock ticks. As a result, different GEM5 instances modeling different systems and applications will require varying amounts of time (as in wall-clock time) to reach certain application milestones and the notion of actual time in those systems will be different. If these systems are connected through a network, communication through actual network protocols cannot be achieved, as ordering or other time-related functions cannot be accomplished.

B. Network Simulator

As mentioned, GEM5 can provide system simulation up to the level of the Network Interface Card (NIC). COSSIM employs a dedicated network simulator (OMNET++ [2]) that handles all network related modeling from the physical layer of a NIC and beyond. As such each simulated node is consisted of two parts, the processing and the network simulation one communicating through the HLA framework. Inside OMNET++, these nodes are called HLA Enabled nodes and they can co-exist in the same simulation with conventional OMNET++ nodes. This way, COSSIM supports modeling of network devices such as bridges, switches and routers.

Each of the HLA Enabled Nodes has a minimum functionality that allows them to communicate with their counterpart on the GEM5 side. This communication is established through an HLA run-time infrastructure (RTI) wrapper. More specifically a CERTI-HLA compliant wrapper was developed, offering a unique interface to each node simulated in the network subsystem to communicate consistently and synchronized with the processing subsystem of the COSSIM simulator. These nodes communicate (via the HLA sockets) transparently with the processing simulator. All the added functionality was deployed in the user space to assure 100% compatibility with OMNeT++ and its adopted libraries (e.g INET).

In order for the COSSIM simulator to increase the simulation accuracy the upper protocol stack of the network should be simulated in the processing sub-system (that is GEM5). However the network sub-system should be able to forward network packets in the data link layer (L2) or in the network layer (L3) so that other aspects of the network to be modelled (e.g packet latency). OMNeT++ does not send the actual payload data (from the application layer) across the simulated network to gain performance. In the case of COSSIM, since the goal is to provide cycle-accurate simulation, the whole protocol stack is executed, thus producing a fully RFC-compliant binary IP packet. COSSIM extends OMNET++ through a custom en/de-capsulation process that ensures that RFC-compliant binary packets can be properly handled both by the processing nodes running standard OSes and internally in the network simulator, so that all functionality of the simulator and its models is preserved.

C. Towards a Consistent Framework

To provide a consistent framework that connects all pieces together and presents itself as a single simulator to the user, COSSIM implements proper synchronization mechanisms that ensure a common notion of time throughout all GEM5 instances and OMNET++, as well as a GUI that enables the user to easily orchestrate a simulation.

COSSIM simulator synchronization is achieved through CERTI HLA in two stages: *Synchronization per node* and *Global Synchronization*. The first one ensures proper communication between each simulated node (through GEM5) and its representation in the network model formed in OMNET++. The latter is required so that all nodes that participate in the simulation are synchronized. This is required as COSSIM supports different types of CPUs with potentially different clock cycles and/or different network protocols, all resulting in varying workload for the simulators engine. Therefore, the simulated time (the timing of the modelled system) in each node can be completely different given the same real-time (the simulation time). Through Global Synchronization a unified notion of time is achieved.

Concerning the GUI, it is based on an extension of the GUI provided by OMNeT++ that integrates all components of the framework. The GUI acts as a kind of wizard that guides the user through the GEM5 configuration process for each of the simulated nodes as well as the modeling of the network. Besides reducing the effort to setup and initiate a simulation, the wizard prevents the user from setting wrong various parameters and thus minimizing risk of starting a time-consuming simulation that will latterly be proven erroneous.

III. Demonstrating COSSIM: Mobile Visual Search Cloud Application

Mobile Visual Search [5] (MVS) is a computer vision application built on the idea of retrieving interesting information about physical objects using only image content; multimedia content can be send back in response to a search or to a user's action. The aim of MVS is to analyze a query image taken by the user and search for similar images inside a large database. MVS is already used in many different applications such as interactive museum guides, e-commerce mobile applications, localization systems and many more.

A. Algorithm overview

The MVS algorithm is composed of two consecutive modules: the Image Analyzer (IA) and the Retrieval Stage (RS).

Firstly, in the IA, a series of advanced image processing techniques are applied on the image acquired by the user in order to localize relevant region of it (key-points). Each selected key-point is then analyzed and, for each one of them, a local descriptor is extracted. The last operation of the IA is the aggregation of the computed local descriptors into a compact global descriptor which describes the whole image. These descriptors are concatenated and further compressed in order to reach a size of around 15-20 KB for each VGA image. The whole IA stage can be performed directly by the user's mobile device.

On the other hand, the RS is able to compare the image descriptor created in the IA with a plethora of other descriptors extracted off-line from each image in the database. In the first step of RS, the images in the database are quickly ranked in respect to the similarity of their global descriptor in respect to the one computed from the query image. In this way only a short list of the images in the database are selected as possible matches. A more precise comparison using local descriptors, including geometry checks and outliers rejection, is performed and the most similar image to the query one is finally selected. The RS requires access to the whole database and for this reason it is performed server-side.

B. System Specifications

1) Native system: The basic components of a client/server system able to run a MVS application are the following:

- Client Device: typically the device that is employed is a smartphone class device or more generally a device that provides a means to capture a digital image and a processing system (application processor) that can perform basic computational tasks on the image. For the specific evaluation of the COSSIM framework, digital imaging sensors have not been considered and each image is simply read from file. We tested the MVS application on two different devices: Hardkernel Odroid XU3 (equipped with Cortex-A15 at 2.0 GHz quad core and Cortex-A7 1.4 at GHz quad core CPU) and STMicroelectronics B2260 [6] (a dual Cortex-A9 at 1.5 GHz). For both of them a single core (A7 and A9 respectively) with maximum working frequency set as 1 GHz, has been selected for comparison reasons.
- Server: The server uses an Intel Core i5-5200 (2.2 GHz) quad core processor.
- Communications: Wireless links between the server node and the client SoCs/imaging nodes are assumed in order to send the visual search queries to the cloud and receive the appropriate responses.

2) Simulated system: The MVS cloud application simulated inside COSSIM is composed as follow:

- 1 single core imaging node (GEM5 ARM with 1 GHz CPU, 512 MB RAM) running the Image Analyzer.
- 1 central node / server (GEM5 x86 with 2 GHz CPU, 512 MB RAM) running the Retrieval Stage.
- Wireless network for communication.

TABLE I
GLOBAL AND LOCAL MATCHING SCORES ON THE BUILDING DATABASE. THE RESULT IMAGES CORRECTLY DEPICT THE SAME OBJECT AS THE QUERY ONE IN BOTH SIMULATED AND NATIVE SYSTEM.

Query	1st result	2nd result	3rd result
			
Native			
Local Score	93.72	60.21	13.06
Global Score	315.759	63.202	50.172
Simulated			
Local Score	93.72	60.21	13.06
Global Score	315.759	63.202	50.172

TABLE II
SIMULATED AND NATIVE TIME EXECUTION (IN MS) OF THE MVS APPLICATION ON THREE DIFFERENT IMAGES. HOST TIME IS ALSO REPORTED.

Times (ms)	Simulated	Native A9	Native A7	Host
Image 1	14499.1	18066	15613.7	10941930
Image 2	21459.3	27167.2	23229.2	16133510
Image 3	14437.14	17446.5	15855.9	10750010
Average	16798.63	20893.23	18232.93	12608483

C. Simulation results

The MVS application was applied on the same images inside and outside of the simulation. Table I report the local and global descriptors scores returned by the MVS application applied on the query image reported. As we can see COSSIM is extremely precise and we obtained the same result on the native (A7 and A9 lead to the same result) and simulated system. From Table II we can see that the simulated time is very close to the Native A7 case. The average host time (the time required for running the simulation) is about 3 hours and 30 minutes.

ACKNOWLEDGEMENT

This work is supported by the European Commission in the context of the H2020 COSSIM (Novel, Comprehensible, Ultra-Fast, Security-Aware CPS Simulator) project (#644042).

REFERENCES

[1] The gem5 simulator. [Online]. Available: http://gem5.org/
[2] Omnet++ discrete event simulator. [Online]. Available: https://omnetpp.org/
[3] H. Labs. Mcpat an integrated power, area, and timing modeling framework for multicore and manycore architectures. [Online]. Available: http://www.hpl.hp.com/research/mcpat/
[4] Certi project. [Online]. Available: http://savannah.nongnu.org/projects/certi
[5] M. Paracchini, E. Plebani, M. B. Iche, D. P. Pau, and M. Marcon, "Embedded real-time visual search with visual distance estimation," in *Image Analysis and Processing - ICIAP 2017 - 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings, Part II*, 2017, pp. 59–69.
[6] [Online]. Available: https://www.96boards.org/documentation/ConsumerEdition/B2260/GettingStarted/