COSSIM: An Open-Source Integrated Solution to Address the Simulator Gap for Systems of Systems

Andreas Brokalakis[†], Nikolaos Tampouratzis[*], Antonios Nikitakis[†], Stamatis Andrianakis[*], Ioannis Papaefstathiou[†], Danilo Pau[‡], Emanuele Plebani[‡], Marco Paracchini[§], Marco Marcon[§], Ioannis Sourdis[¶], Prajith Ramakrishnan Geethakumari[¶], Maria Carmen Palacios[‖], Miguel Angel Anton[‖], Attila Szasz[**]

[*]Telecommunications Systems Institute, Technical University of Crete, Greece
[†]Synelixis Solutions Ltd, Chalkida, Greece
[‡]Advanced System Technology, STMicroelectronics, Agrate Brianza, Italy
[§]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy
[¶]CSE Department, Chalmers University of Technology, Gothenburg, Sweden
[‖]Tecnalia Research and Innovation, San Sebastian, Spain
[**]SEARCH-LAB, Budapest, Hungary

***Abstract*—In an era of complex networked heterogeneous systems, simulating independently only parts, components or attributes of a system under design is not a viable, accurate or efficient option. The interactions are too many and too complicated to produce meaningful results and the optimization opportunities are severely limited when considering each part of a system in an isolated manner. The presented COSSIM simulation framework is the first known open-source, high-performance simulator that can handle holistically system-of-systems including processors, peripherals and networks; such an approach is very appealing to both CPS/IoT and Highly Parallel Heterogeneous Systems designers and application developers. Our highly integrated approach is further augmented with accurate power estimation and security sub-tools that can tap on all system components and perform security and robustness analysis of the overall networked system. Additionally, a GUI has been developed to provide easy simulation set-up, execution and visualization of results. COSSIM has been evaluated using real-world applications representing cloud (mobile visual search) and CPS systems (building management) demonstrating high accuracy and performance that scales almost linearly with the number of CPUs dedicated to the simulator.**

I. Introduction

Nowadays, highly interconnected systems of systems such as Cyber Physical Systems (CPS), Internet of Things (IoT), parallel and distributed systems (e.g. Cloud and HPC systems) are growing in capability at an extraordinary rate, incorporating computing systems that vary from simple microcontrollers to high performance units connected with each other through a multitude of networks. One of the main problems designers of such systems face is the lack of simulation tools that can offer realistic insights beyond simple functional testing, such as the accurate performance estimation, precise timing, power/energy estimations and network deployment issues. On top of that, none of the existing solutions provide security testing to assess potential design problems.

In this paper, we present the COSSIM Simulation Framework, an open-source simulation framework for systems of systems that aims to address all the aforementioned limitations. The proposed solution integrates a series of software packages that model the computing devices of the processing nodes and the network that interconnects them. It provides cycle accurate results by simulating the actual application and system software executed on each node, realistic communication modeling of the different networks that are employed and power/energy consumption estimates for both the processing elements and the network based on the actual dynamic usage scenarios. The simulator provides the necessary hooks to security testing software, making it possible to determine vulnerabilities and examine the robustness of the system under design. Last but not least, by employing a standardized interconnection protocol between its components (IEEE 1516.x HLA), the framework can be extended to include additional tools, such as simulators of physical processes.

II. Related Simulation Tools

The COSSIM simulator applies to system of systems in general. In that context, two main classes of systems may be considered with a number of simulation tools available for each one. The first class encompasses systems that are more broadly known as Wireless Sensor Networks (WSN), Cyber-Physical Systems (CPS) and Internet of Things (IoT), while the latter includes Cloud and parallel/distributed systems.

Starting from the first domain, the available simulation tools fall under two main categories. The first focuses mostly on the functionality of the system under design and thus tries to model different entities in the system: a physical process, an electromechanical component, user behavior, events, message exchanges etc. Such tools are mostly based on the use of well-defined models of computation. Among them the most profound ones are Ptolemy [1], Matlab Simulink [2] and Modelica[2]-based simulation environments. These tools can only be used during the initial design phase of an application as they

[2]Modelica is a language used to construct models of systems. Multiple simulation environments that can import Modelica models exist such as OpenModelica, JModelica, CyModelica.

978-1-5386-7377-5/18/$31.00 ©2018 IEEE
DOI 10.1109/DSD.2018.00033

cannot handle cycle-accurate processing simulation, power consumption estimations of the actual processing components nor the simulation of the actual networks that are going to be employed.

Simulators that do handle those aspects can typically be found in the WSN field. Most of these tools are designed around a specific WSN-oriented operating system or a specific device. For example, TOSSIM [3] can simulate WSN applications designed for the TinyOS operating system. It can simulate the whole OS stack including application software, while offering models for specific micro-controllers. Similarly, COOJA [4] is built to simulate sensor networks whose nodes run mainly the Contiki OS, and it too can only support simple micro-controllers and limited network protocols. Other WSN simulators [5], [6] focus on specific devices or motes (they can model the whole sensor node including sensors and the radio communications chip). These simulators can effectively model both the actual software executed on the processing nodes and the network between the devices, however their applicability cannot be generalized neither in terms of software (e.g. support for different OSes or software packages), nor in terms of hardware (e.g. support for other CPUs) or network (they can only simulate a specific wireless protocol). As a result, those tools cannot handle the simulation of general interconnected systems that require the modeling of very diverse processing devices together with a multitude of networks.

In the area of Cloud simulators the most widely used simulation framework is CloudSim [7] and its numerous extensions or derivatives. These simulators are working at a high abstraction level using generic models and thus they cannot precisely simulate the actual execution of an application neither can they model the exact hardware used. Other cloud simulators focus on specific aspects of the data center that are useful for cloud providers such as resource provisioning. Those simulators model user and application requirements through stochastic processes or mathematical models to predict resource usage and provide optimization insights for avoiding excessive overprovisioning and underutilization (e.g. [8]).

Complimentary to the cloud data servers, while also sharing certain aims and features, are the parallel systems intended to execute highly complex and demanding applications (HPC). There are certain tools such as [9] which model in a cycle accurate manner processing units, while others (e.g. [10], [11]) that focus on the simulation of real networks. As such they either focus on providing highly accurate results per node and more simplistic network models or they replace the cycle-accurate simulation with application traces or functional modeling and focus on the network part.

III. COSSIM Framework

From the previous section, it becomes evident that no integrated solution exists that can handle the simulation of an actual system of systems, including its complete software stack, network dynamics and energy aspects. COSSIM aims to fill this gap by forming a tightly integrated framework as a set of well-established tools. The goal is to provide a single solution (as if it were a single tool rather than a framework) that offers functionality greater than what can be achieved by using each component separately.

A. Processing Simulator

A system of systems is comprised of a set of nodes and a number of networks that connect them together. In designing a general-purpose simulator, the diversity of nodes can be large; from simple sensor nodes to server systems. Thus, to accurately simulate a node that can vary as much, a system simulator that is cycle-accurate, Instruction Set Architecture (ISA) independent, configurable (in terms of supported devices and system features), able to boot real-world operating systems and execute software compiled for them is required.

The simulation tool that can cover these aforementioned requirements as closely as possible is GEM5. GEM5 [12] is a computer system simulation platform that can cover almost all the aforementioned requirements. It 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 be configured to support a broad range of ISAs. While GEM5 does not support natively any power/energy estimations, in COSSIM it is integrated with McPAT [13] to provide such information.

B. Network Simulator

GEM5 can provide system simulation up to the level of the Network Interface Card (NIC). It does not support network modeling[3]. Therefore, COSSIM employs a dedicated network simulator that handles all network related modeling from the physical layer of a NIC and beyond. The simulator chosen is OMNET++ [15]. Through OMNET++ different network protocols and topologies can be supported and a realistic network behavior of the system can be modeled (devices such as bridges, switches, routers that are part of the infrastructure rather than the main system developed can also be modeled to increase accuracy). Through OMNET++'s extensions, it is also possible to estimate the power consumed at the network as well as the radio devices of the nodes (MiXiM add-on [16]).

C. Integration of Components

Binding the aforementioned processing and network simulators together is not trivial, as it requires carefully designed communication interfaces and synchronization schemes. These bidirectional interfaces have to pass information on the type and timing of events and to provide a common data representation throughout the framework. To achieve this task, COSSIM employs the IEEE High Level Architecture (HLA) [17]. HLA is a general purpose software architecture specifically designed for the development and execution of distributed simulation applications, determining the functional entities, design rules and interfaces for simulation systems and specifying the communication between individual components. It requires a

[3]It should be noted that a new effort, dist-gem5 [14] tries to fill this gap, however network support and configuration is very limited and cannot be used for general network structures

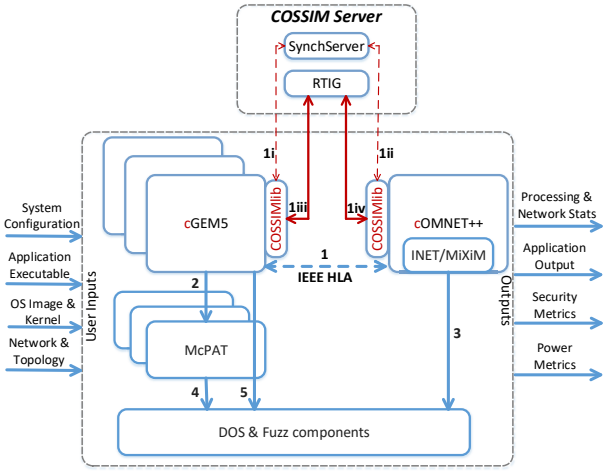

Fig. 1. Top-level view of the COSSIM framework

runtime infrastructure (RTI) to perform tasks such as synchronization and simulation control. Among the numerous RTI implementations, COSSIM has adopted CERTI [18].

IV. Implementation

Figure 1 demonstrates the COSSIM simulator with all its components and interfaces. Multiple instances of a node simulator module (GEM5) cover the processing nodes of a system (each instance simulates a different node). The networks that binds together all nodes are simulated by the network module (OMNET++). The GEM5 instances are connected with the OMNET++ simulator through IEEE HLA compliant interfaces (Interface #1). Additionally, each GEM5 instance is connected with a McPAT instance to estimate the energy/power consumption of each node. Both GEM5 instances and OMNET++ have been properly modified to provide hooks that allow security software components (custom-built within COSSIM) to interact with the simulation process so that security tests can be performed (interfaces #3, #4, #5).

Figure 1 also illustrates the primary inputs that a user has to provide to the overall system and the primary outputs of the simulation process. Specifically, *System Configuration* (number of CPUs, memory sub-system, peripherals etc.) have to be defined either using a configuration file or the supplied graphical user interface. Furthermore, *image files* of the OSes that will be executed on the nodes of the simulated platform have to be provided. These include the OS kernel, the *application executable* and all the libraries that are required. Finally, through a *Network & Topology description*, the user can define a network topology and describe channel definitions and network protocols.

Upon completion of a simulation, the COSSIM simulator produces a number of outputs. *Processing & Network* statistics contain all statistics of an execution run (clock ticks, real time execution, number of packets sent or dropped, delay of received packets, peak throughput, etc.). The output of the actual application that is executed is also provided. When security tests are performed, *security metrics* (increase in response time, rejection rate, vulnerability density, system robustness, etc.) are reported as well. Finally, power and energy estimations for each node (including further details such as peak power, runtime dynamic power, leakage etc) and for the network are provided.

A. GEM5 Adaptation for COSSIM Integration

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. GEM5 supports networking through a virtual dummy link (Etherlink) 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 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 they have to be identical (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 them at the same time to match the specific network protocol required by the application (Ethernet, WiFi, 3G, etc). To achieve this interconnection with the network simulator, CERTI HLA interfaces have been employed. A custom library (*COSSIMlib*) has been integrated in GEM5 through Etherlink that exchanges Ethernet Packets captured from the Etherlink Device and sends (and accordingly receives) them to (from) the HLA Server. The HLA Server forwards these messages to a proper interface of a Network Simulator that implements all the network related functionality.

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 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, 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. OMNET++ Adaptation for COSSIM Integration

As mentioned, GEM5 can provide system simulation up to the level of the Network Interface Card (NIC). COSSIM employs a dedicated network simulator (OMNET++ [15]) 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 is achieved through the HLA runtime infrastructure, that provides a unique interface to each node simulated in the network subsystem to communicate consistently and synchronized with the processing subsystem.

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 (cGEM5). 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 can be modeled (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/decapsulation process that ensures that these 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. Power Estimation and Security Tools

McPAT is integrated with GEM5 to produce power and energy estimations for each simulated node. In COSSIM a number of changes within McPAT have enabled the tool to produce more accurate results and extended it to be able to work with GEM5 for the simpler CPU models. Furthermore, since McPAT may be invoked multiple times from GEM5 during a simulation run (e.g. during security tests), an FPGA-accelerated version of the tool has also been produced, effectively minimizing the cost of using a power estimator.

Additionally, COSSIM integrates tools that allow the evaluation of the security of the system simulated. The security module consists of three main parts; the *DoS Testing System* (DTS), the *Fuzz Testing System* (FTS) and the *Metrics Management* (MM). The DTS tests the resilience of a system under simulation against various types of denial-of-service attacks. The FTS provides automated testing of component interfaces for discovering vulnerabilities. It produces input test vectors that are fed in the application under test. This process (fuzzing) is capable of exposing errors that arise as a result of processing these input vectors and is also used for quality assurance to evaluate a systems robustness. The MM observes the state of the simulated system to calculate various types of security metrics, allowing the systems developers to assess its behavior in certain types of situations.

D. 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 ensures proper communication between each simulated node and its representation in the network model formed in OMNET++. This type of synchronized communication is necessary because network data between the two simulators must be exchanged while preserving the exact same time ordering.

Global Synchronization is required so that all nodes that participate in the simulation remain synchronized. Since COSSIM supports different types of CPUs with potentially different clocks, these result in varying workload for the simulators engine. Therefore, the simulated time (the timing of the modeled 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.

The *Synchronization time per node* and the *Global Synchronization time* are two different entities that can be separately defined by the user. The first one is mostly defined by the latency of the network interface and it doesn't constrain the simulation speed while the second is a trade-off between simulation speed and simulation accuracy.

V. Validation and Performance Analysis

The COSSIM Simulation Framework has been validated using two real-world applications with very different requirements and characteristics. The first one is a mobile Visual Search application and the second is a Building Management System. Scaling and performance testing of the simulation framework have been carried out through a synthetic benchmarking process.

A. Mobile Visual Search

Mobile Visual Search [19] (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.

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

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

In the IA stage, a series of advanced processing techniques are applied on the image acquired by the user in order to produce a compact global image descriptor. The whole IA stage can be performed directly by the user's mobile device. The MVS application has been tested assuming two different devices: Odroid XU3 (equipped with a quad Cortex-A15 and a quad Cortex-A7) and STMicroelectronics B2260 (a dual-core Cortex-A9). 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.

The RS compares the image descriptor created in the IA with a plethora of other descriptors extracted off-line from each image in a database. The images in the database are quickly ranked in respect to the similarity of their global descriptors. This way only a short list of the images are selected as possible matches. A more precise comparison using local descriptors is performed to select the most similar image. The RS requires access to the whole database and for this reason it is performed server-side. A server using an Intel Core i5-5200 quad core processor has been used. The server and the mobile nodes are connected through a wireless link.

The MVS application simulated inside COSSIM is composed by a single core imaging node (GEM5 ARM with 1 GHz CPU, 512 MB RAM) running the Image Analyzer, a central node / server (GEM5 x86 with 2 GHz CPU, 512 MB RAM) running the Retrieval Stage and a wireless network (802.11) for communication. Results have been compared with the native platforms mentioned above. 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 (which is the platform most closely modeled by COSSIM).

B. Building Management System

The BMS application is actually deployed in a real smart building environment on the KUBIK building laboratory of Tecnalia. The KUBIK facilities allow the installation and monitoring of a wide range of devices and energy efficiency control systems. The main purpose of the BMS is to deliver the appropriate set-points to the HVAC system to ensure the boundary conditions in each room. Thus, the BMS performs complex energy simulations using environmental sensor measurements and building modeling data that take into account the influence of the thermal inertia of the structure.

TABLE II
SIMULATED AND NATIVE EXECUTION TIME (MS) OF THE MVS 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

The benefits of using the COSSIM simulator to analyse the operation of KUBIK's BMS come from the necessity of optimising both the behavior of network and processing performance while ensuring the resilience of the system against security attacks. The COSSIM framework evaluated the BMS of KUBIK in the following different scenarios:

- Performance and network evaluation of BMS using a complex sensor network. This case includes forecasting processing and wired sensor communication at the same time. Both regular and burst traffic situations were tested.
- Security evaluation of BMS under Man-in-the-Middle (MitM) attack. This case evaluates the BMS resilience under active MitM attacks focusing on confidentiality and integrity of the exchanged data and the availability of the server node.

The setup of the evaluation included four nodes, one representing an x86-based server system and the other three were ARM-based sensor nodes. The sensing nodes collect environmental data, while the server system performs a simplified local forecasting using a neural-network based module. The use of COSSIM with pre-collected sensor data demonstrated that the simulated system provided the same responses with the actually deployed system. Additionally, the ability to perform security tests resulted in altering the installed system software in the KUBIK building to address the issues discovered.

C. Performance Scaling

The key concept of COSSIM's approach is to execute the OMNET++ simulator in a workstation due to the GUI that facilitates the orchestration and visualization of the simulation, while the GEM5 instances are run in one or more servers (distributed simulation). For this reason, OMNET++ simulation was carried out on a PC based on a quad core Intel CPU (Machine 1). Simulation of GEM5s as well as SynchServer with HLA Server were run on a server based on two Intel Xeon processors with 12 CPU cores in total (Machine 2). To demonstrate the performance scaling a network of ARM or X86 nodes is formed and the overall simulation time required so that each node boots in a full Linux OS is measured.

Figure 2 demonstrates the impact of the synchronization interval in the performance of the system, as measured by the overall (wall-clock) time required to complete a simulation

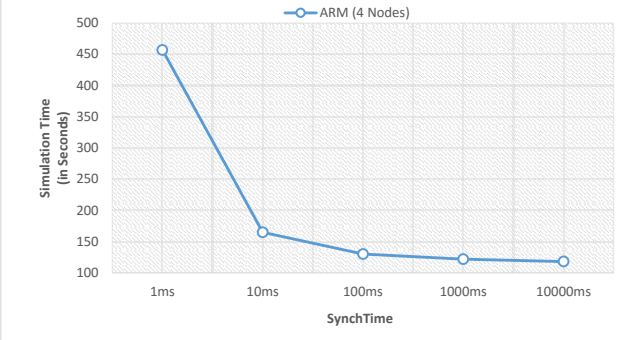

Fig. 2. COSSIM simulation time using different synchronization intervals

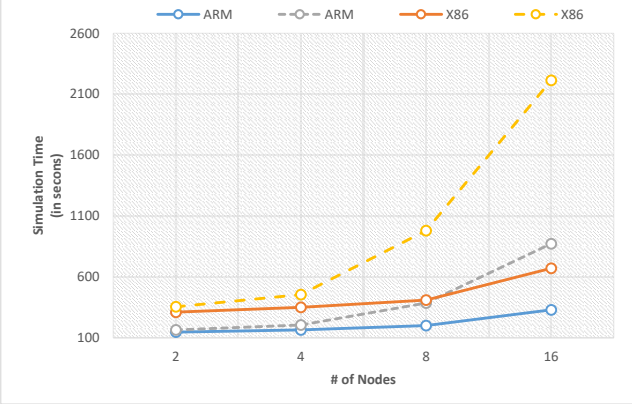

Fig. 3. COSSIM simulation time (dashed lines describe the case of all GEM5s and OMNET++ residing on the same physical machine, solid lines are for the distributed simulation scenario)

with 4 nodes (all configured for ARM ISA and the overall framework is executed on Machine 1). There is a dramatic drop in performance for very short interval times (smaller than 10ms), since beyond that limit the number of synchronization events becomes excessive and practically halts the execution of the different components too frequently.

Figure 3 demonstrates how performance scales as more resources are made available to the simulator (here a synchronization interval of 10ms is chosen). Practically, when the number of physical CPU cores matches the number of nodes in the simulated system, the performance degradation is minimal. Once the number of nodes exceeds the available CPU cores, there is a sharp performance drop. Additionally, one may observe that the impact of the network (this is a distributed simulation scenario) is limited.

VI. Conclusions

In this work, we present COSSIM Simulation Framework, an open-source tool that aims to address in a single integrated solution both the processing and the network part of a system of systems while taking into account security and power aspects of the nodes. By using a novel synchronization scheme it always assures accurate simulation, distributed processing and extensibility. Validation results and performance measurements demonstrate that it produces functionally correct results, accurate performance estimations and it can scale effectively making it feasible to simulate large systems.

References

[1] The ptolemy project. [Online]. Available: http://ptolemy.eecs.berkeley.edu/

[2] Model-based design of cyber-physical systems in matlab and simulink. [Online]. Available: https://www.mathworks.com/discovery/cyber-physical-systems.html

[3] Tinyos simulator. [Online]. Available: http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM

[4] Cooja simulator. [Online]. Available: http://anrg.usc.edu/contiki/index.php/Cooja_Simulator

[5] Avrora - avr simulation and analysis framework. [Online]. Available: http://compilers.cs.ucla.edu/avrora/

[6] Worldsens: Development and prototyping tools for application specific wireless sensors networks. [Online]. Available: http://wsim.gforge.inria.fr/tutorials/wasp/files/wsim-tutorial.pdf

[7] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, 2011.

[8] D. Meisner, J. Wu, and T. F. Wenisch, "Bighouse: A simulation infrastructure for data center systems," in *Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on*. IEEE, 2012.

[9] M. Hsieh, K. Pedretti, J. Meng, A. Coskun, M. Levenhagen, and A. Rodrigues, "Sst+ gem5= a scalable simulation infrastructure for high performance computing," in *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012.

[10] W. Hurst, S. Ramaswamy, R. Lenin, and D. Hoffman, "Development of generalized hpc simulator," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2010.

[11] C. Minkenberg, W. Denzel, G. Rodriguez, and R. Birke, "End-to-end modeling and simulation of high-performance computing systems," in *Use Cases of Discrete Event Simulation*. Springer, 2012.

[12] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[13] 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/

[14] A. Mohammad, U. Darbaz, G. Dozsa, S. Diestelhorst, D. Kim, and N. S. Kim, "dist-gem5: Distributed simulation of computer clusters," in *Performance Analysis of Systems and Software (ISPASS), 2017 IEEE International Symposium on*. IEEE, 2017.

[15] Omnet++ discrete event simulator. [Online]. Available: https://omnetpp.org/

[16] Mixim simulator. [Online]. Available: http://mixim.sourceforge.net/

[17] Ieee 1516-010 - standard for modeling and simulation high level architecture - framework and rules. [Online]. Available: https://standards.ieee.org/findstds/standard/1516-2010.html

[18] Certi project. [Online]. Available: http://savannah.nongnu.org/projects/certi

[19] 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.