Cunetsim: A New Simulation Framework for Large Scale Mobile Networks* †

Ben Romdhanne Bilel Nikaein Navid
Eurecom, 2229, route des Cretes, BP 193
F-06560 Sophia-Antipolis cedex, France
{benromdh,nikaeinn}@eurecom.fr

ABSTRACT

Most of the existing packet-level simulation tools are designed to perform experiments modeling small to medium scale network nodes. The main reason of this limitation is the amount of available computation power and memory in quasi mono-process simulation environment. To enable efficient packet-level simulation for large scale scenario, we introduce a CPU-GPU co-simulation framework where synchronization and experiment design are performed in CPU and node's logical processes are executed in parallel in GPU according to the master/worker model. The framework is developed using Compute-Unified Device Architecture and denoted as Cunetsim, CUDA network simulator. In this work, we study the node mobility and connectivity as they are among the most time consuming task when large scale network is simulated. Simulation results show that Cunetsim execution time remains stable and that it achieves significantly lower execution time than existing approach when computing mobility and connectivity with no degradation in the accuracy of the results.

Categories and Subject Descriptors

H.4 [**Techniques for network measurement, simulation, and emulation**]:

General Terms

Large Scale Simulation, GPGPU, Distributed Simulation

Keywords

Large Scale, Simulation, Evaluation

*(Produces the permission block, and copyright information). For use with SIG-ALTERNATE.CLS. Supported by ACM.

†A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2ε and BibTeX* at `www.acm.org/eaddress.htm`

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WOODSTOCK '97 El Paso, Texas USA
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$10.00.

1. INTRODUCTION

Packet-level simulators are usually based on a discrete event paradigm where sequences of events are generated and processed. In general, such events represent mobility and connectivity calculation, protocol operation, and in/out packet processing. The time complexity of a simulation is then proportional to number of nodes and number of packets scheduled to be processed, which represent two main bottlenecks when targeting scalability. To enable efficient and scalable simulation environment, parallel node execution environment with minimal inter-process communication overhead are needed to improve simulation performance when total number of nodes and packets increase significantly [6].

The traditional approach to deal with large scale simulation is CPU parallelism and distributed simulation [8]. SwimNet is on example of distributed parallel simulation environment designed for large scale wireless network [4]. It is based on a master process which configures and runs distributed simulation. This approach introduces a significant overhead due to the synchronization among different processes and machines and requires sophisticated and expensive simulation infrastructure [7]. Furthermore, mobile network requires periodic node and link updates, which needs to access global information such as nodes' position across multiple machines causing distributed simulation to be inefficient in terms of execution time. Sinalgo is an example of a mono-process simulator that is capable of simulating up to one hundred thousand nodes [1]. It only focuses on the verification of network algorithms, and provide a message passing view of the network by abstracting the underlying layers. The performance degradation happens when the number of nodes increases as it requires the global view of the network prior to each iteration.

This work presents an efficient and scalable CPU-GPU co-simulation framework for testing and validating network algorithms, denoted as Cunetsim, CUDA Network Simulator. As opposed to previous works, Cunetsim is designed to provide an independent parallel execution environment for logical processes of a node. Nodes communicate with each other only through the message passing based on the buffer exchange, thus avoiding the usage of any global knowledge. Furthermore, it exploits the master/worker model for CPU-GPU co-simulation and provides CPU-based conservative synchronization using GPU clocks.

The remainder of the paper is organized as follows. Section 2 briefly presents the concept of Cunetsim and its design features. Preliminary results are given in section 3 followed by concluding remarks and future directions in section 4.

2. CUNETSIM DESIGN

Cunetsim is designed based on three models: (i) experiment model, where a sequential experiment model is employed to allow a modular and reproducible simulation, (ii) node model, where each function/service of a node is performed on an independent logical process, (iii) simulation model, where the master/worker parallel discrete event model is used between CPU and GPU within the same OS context. In the following subsection the models are explained.

2.1 Experiment Model

An experiment is modeled as a sequential experiment workflow, where the output of each step will be the input of the next, to allow a modular and reproducible description of a simulation scenario. Five consecutive steps are defined: scenario description, configuration, execution, monitoring, and analysis [2]. In Cunetsim, the execution is performed in the GPU context, and the remaining steps in the CPU context as they are not representing intensive processing. However, depending on the processing load and required simulation interaction, a given step may be switched between CPU and GPU context.

2.2 Node Model

A node is modeled as a stack of independent logical processes (LP) each of which performs a specific functionality/service on behalf of a node. Nodes communicate with each other only through the message passing (i.e. restrictive communication). Only buffers are exchanged between nodes to avoid global knowledge and centralized information.

In Cunetsim, each node contains four ordered LPs: mobility (MOB), connectivity (CON), protocol (PROTO), and packet (PKT). Each LP has its own container (i.e. GPU kernel) and executed in parallel. For each node, the container will be periodically launched and select corresponding data and model applicable to each LP based on the kernel unique identifier (KID). Nodes are logically located in a geometric area called cell used by MOB and CON processes to determine the network topology. The MOB calculates the movement in a space following the mobility model and moving dynamics (e.g. min and max speed) of a node. It implements several mobility and boundary policy models. The current version supports two mobility models: random way point and random walk, and three boundary policy models: annulment of excess, sliding on the boundaries, and bouncing on the boundaries [3]. The CON determines network topology over time following the connectivity model and nodes' position. It counts the number of nodes in each cell and updates the nodes' ID and positions to build the network connectivity according to a set of models including channel, interference, power, transmission models. Currently, only two simple channel models are supported: unit disk graph and quasi unit disk graph. The PROTO describes network nodes and their operation and algorithm. It implements methods that are called when a node sends and receives a message (i.e. protocol data unit). The current version implements different broadcasting schemes. The PKT encapsulates protocol messages in packets, which contains meta information to perform the message delivery, and provides all the actions associated with the packet processing when nodes communicate with each other. Packets are exchanged through in/out buffers following the connectivity model, and the packet reception is determined at the receiver side.

2.3 Simulation Model

The simulation is modeled based on a master/worker simulation model[8], where the master is a CPU process and workers are GPU (and in some cases CPU) threads within the same OS context. Due to this specificity, the management of master and workers interaction can be simplified. The main simulation model is designed around node and LPs. Each node is composed of several LPs and each LP is implemented in a unique process P and executed sequentially (see Figure 1a). The master process resides in CPU context and is responsible for global time synchronization and the simulation coherence, which is achieved through safe timestamp-ordered processing of simulation events within each LP, also known as conservative synchronization [9]. This ensures that an LP does not execute an event until it can guarantee that no event with a smaller timestamp will later be received by that LP.

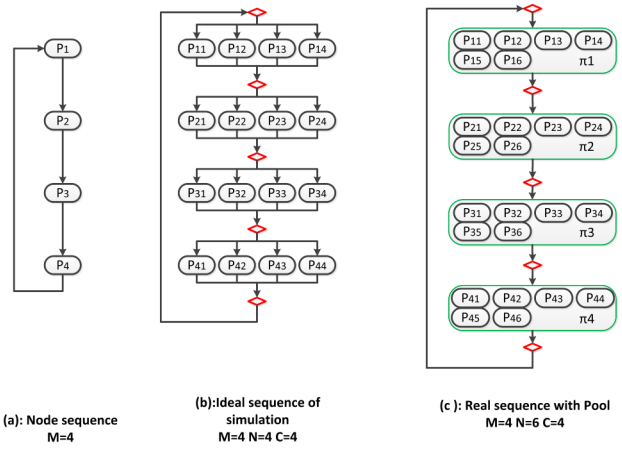

Figure 1: Node LP sequence and pool

In a general case, each node is composed of M processes and the experiment evolves N nodes. A simulation round is defined by the execution of all process of all nodes one time which means that the total number of processes in one round is $M * N$. In ideal conditions, the N nodes will be executed in parallel and their processes sequentially as shown in (figure 1 b). Now assume that only C cores (or GPU kernels) are available, which implies that only C nodes could be executed in parallel. In this case, the correctness of simulation will be preserved if and only if : $\forall i \in [0, M] \wedge j \in [0, N]$, $P_{i+1,j}$ could not start until all $P_{i,j} \forall j$ ends. To achieve this, a sequence of process pool Π_j is defined incorporating the same process for all nodes (see Figure 1c). For a given Π_j, all P_{ij} processes must ends to assert that Π is achieved. This presents a simple yet efficient implementation of the coherence and consistency paradigm [5].

3. PERFORMANCES EVALUATION

We compare the performance of Cunetsim with Sinalgo in terms of total execution time for computing node mobility and network connectivity over increasing the number of nodes. We vary the total number of simulated nodes from 256 to 64k, and apply the random way point (RWP) mobility model and unit disk graph (UDG)connectivity model to measure the execution time for each computation. The

experimentations are done using the same scenario and hardware (i7 940 CPU with 4GB RAM and NVIDIA GTX 460 SE GPU with 1GB GDR5). In order to assess the performance gain and simulation efficiency when GPU is used, Cunetsim is also built on the CPU target and compared with the GPU. In addition, the performance gain of the parallelism approach on CPU target in comparison with the sequential approach is evaluated.

Figure 2 presents the average execution time to compute node mobility in Sinalgo and Cunetsim in GPU and CPU mode. It can be seen that the execution time in Cunetsim is up to seven orders of magnitude lower than Sinalgo when computing the node mobility, and that it remains stable as the number of nodes increases. Compared to Sinalgo, Cunetsim is up to 5 times faster in CPU and 74 times in GPU.

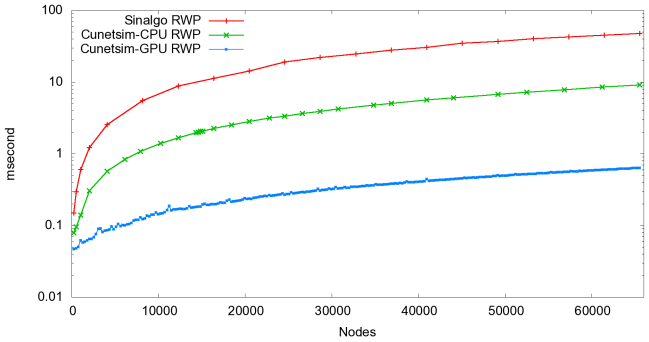

Figure 2: Execution time for mobility

Figure 3 presents the average execution time when computing node connectivity in Sinalgo and Cunetsim in GPU and CPU mode. It can be seen that the execution of Cunetsim remains linear as the number of nodes increases while that of Sinalgo increases exponentially with the number of nodes. In particular, when the number of nodes approach 60k, the execution time of Sinalgo increases drastically. This is because the amount of memory to represent the state of each node exceeds the available memory. In Cunetsim, we observe that execution time remains stable as it exploits the high bandwidth of the GPU memory and its parallel processing power allowing Cunetsim to be radically faster than Sinalgo. When compared with Sinalgo, the performance gain increases with the number of nodes and can reach 80 orders of magnitude higher than Sinalgo for 60k nodes.

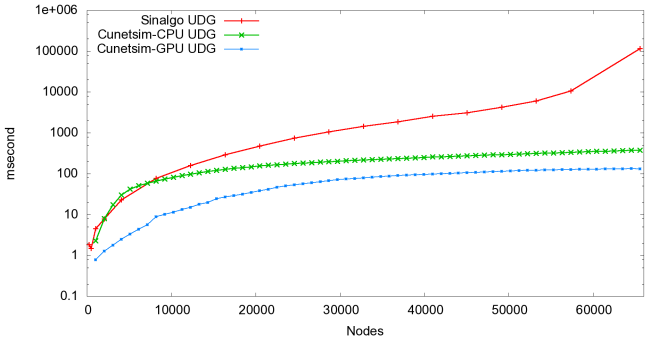

Figure 3: Execution time for connectivity

4. CONCLUSION & FUTURE WORK

New challenges emerge when simulating a large scale mobile network. While network simulation tools are widely used for validation and performance evaluation, their scalability and efficiency remain challenging. Cunetsim aims to unlock the parallel capabilities of the state-of-the-art hardware and software architectures to achieve simulation scalability and efficiency with significantly lower cost. It provides a CPU-GPU co-simulation framework for testing and validating network protocols and algorithms for large scale scenarios. Preliminary results show that the execution time could be radically improved when CPU-GPU parallelism is used.

In the future work, we will evaluate the execution time of Cunetsim with other existing approach such as NS-3 and extend the results for protocol and packet operations. We also plan to further optimize the parallelism capability of Cunetsim by exploiting multi-GPU architecture as a simulation platform.

5. ACKNOWLEDGMENTS

This paper describes work undertaken in the context of the LOLA and CONECT project. The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement no. 248993 and no. 257616.

6. REFERENCES

[1] http://disco.ethz.ch/projects/sinalgo/.

[2] B. Bilel, N. Navid, K. R., and B. C. Openairinterface large-scale wireless emulation platform and methodology. In *MSWIM*, pages 109–112. ACM, 2011.

[3] A. Boukerche and L. Bononi. Simulation and modeling of wireless, mobile, and ad hoc networks. *Mobile ad hoc networking*, pages 373–409, 2004.

[4] A. Boukerche, S. K. Das, and A. Fabbri. Swimnet: A scalable parallel simulation testbed for wireless and mobile networks. *Wireless Networks*.

[5] S. De Munck, K. Vanmechelen, and J. Broeckhove. Revisiting conservative time synchronization protocols in parallel and distributed simulation. 2011.

[6] R. Fujimoto, K. Perumalla, A. Park, H. Wu, M. Ammar, and G. Riley. Large-scale network simulation: how big? how fast? In *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.*, pages 116 – 123, oct. 2003.

[7] A. Park and R. Fujimoto. Efficient master/worker parallel discrete event simulation. *2009 ACMIEEESCS 23rd Workshop on Principles of Advanced and Distributed Simulation*, pages 145–152, 2009.

[8] A. Park and R. M. Fujimoto. Parallel discrete event simulation on desktop grid computing infrastructures. *International Journal of Simulation and Process Modelling*, 5(2):157 – 171, 2009.

[9] K. Perumalla. Parallel and distributed simulation: traditional techniques and recent advances. In *Proceedings of the 38th conference on Winter simulation*, pages 84–95. Winter Simulation Conference, 2006.