Scalability demonstration of a Large Scale GPU-based Network simulator

Ben Romdhanne Bilel; Mohamed Said Mosli Bouksiaa; Nikaein Navid; Bonnet Christian
Mobile Communication Department, Eurecom
{benromdh,mosli,nikaeinn,bonnet}@eurecom.fr

ABSTRACT

Large scale simulation is a challenging issue of the network research area. In particular, simulating one large space where a big number of nodes are in continuous interaction remains complex even if we consider distributed and parallel solutions. In this perspective; GPU appears as a promising hardware providing an important number of independent computing resources. Nevertheless its usage requires a new software design. In that context, Cunetsim is a distributed GPU-based framework which aims to combine the power of GPUs with the flexibility of distributed solution in order to increase the scalability while reducing the complexity. In this work we aim to demonstrate the efficiency and the scalability of that framework on one hand and its robustness in term of event handling on the other hand; therefore we propose a validation scenario including 1.5 millions nodes where we generate up to 10 billions events; we conduct the simulation using one workstation which includes three GPUs.

Categories and Subject Descriptors

I.6.0 [**Computing Methodologies**]: SIMULATION AND MODELING—*General*; C.4 [**Computer Systems Organization**]: PERFORMANCE OF SYSTEMS

Keywords

PADS, PDES, Large scale simulation, System architecture, GPGPU, Heterogeneous computing

1. INTRODUCTION

While large-scale simulations are required to study and validate the behavior of new network technologies and protocols, their establishment is complex and expensive. Moreover, network simulation imposes the modeling of expensive operations such as the mobility, the channel estimation and packets exchange. Thus, increasing the number of node rapidly increases the complexity of those operations which in turn reduces significantly the simulation efficiency . The dominant approach to speedup a large scale simulation is to distribute it over several logical processes (LP) each of which simulates a partition of the main simulated space. Even if such solutions are widely deployed and used in research, the size of each partition was usually limited in term of nodes number and did not exceed thousands of nodes. The major limit remains the computing power required by each LP to handle an important number of nodes. Hence GPU appears as a promising hardware which provides a significant amount of commuting power. In that context, Cunetsim presents a proof of concept that confirms the possibility of running the totality of a network simulation on the GPU.

In this work we aim to focus on the performance of cunetsim as a distributed GPU-based large scale network simulator. The framework design is based on an extended master-worker model that introduces a top level process called *coordinator*. The model is denoted as coordinator-master-worker (CMW). Features and architecture of that framework was discussed in [3]. In particular, we try to highlight the possibility to achieve a large scale distributed simulation based on a network of large networks. Therefore we propose an experimental benchmark which includes 1.5M of nodes distributed over only three active areas (AA). Thus, each AA includes 500K nodes; at the best of our knowledge, this presents the largest contiguous area in term of number of nodes which was simulated on one hardware context (CPU or GPU) .The total number of generated events vary between 4.5 and 10.5 billion; and the event rate spike was about 800 millions events/s. The worst case simulation is done using one workstation which includes 3 GPUs each of which includes 1536 cores. The total runtime for that case was about 410 s which is 13 times faster than the realtime execution.

The remainder of this work is organized as follows. Section 2 presents the GPU background. Section 3 provides an overview of cunestim architecture and features. Section 4 describes the demonstration scenario and setup while evaluation results are in section 5. Finally, section 6 concludes this paper.

2. GPU & GPGPU

Initially dedicated to graphical rendering, Graphics Processing Units (GPUs) are becoming increasingly programmable, flexible and computationally powerful. Modern GPUs are characterized by a high throughput using multi-threading over hundreds of processing cores. GPUs contain independent RAM accessible to all their processing cores which are

Simutools 2013, March 05-07, Cannes, France

DOI 10.4108/icst.simutools.2013.251744

grouped into several streaming multiprocessors (SMs) each of which possessing its specific shared memory. With the venue of General-purpose computing on graphics processing units (GPGPU), we dispose of a user-friendly APIs such as CUDA characterized by a developed ecosystem and a wide range of compliant libraries. For this reason, we choose CUDA as a technological support.

3. CUNETSIM FRAMEWORK

Cunetsim is a distributed GPU-based framework designed for wireless mobile network simulation. It aims to achieve two main goals: enabling extra large-scale network simulation and providing a significant speedup compared to traditional CPU-based frameworks. Therefore, Cunetsim is based on an innovative methodology on the network simulation area[1]. This approach considers that GPUs architecture is adequate to hold the totality of a network simulation based on a CPU-GPU co-simulation where the GPU is the main simulation environment and the CPU is a controller. The scalability in Cunetsim is considered natively since it dedicates an independent execution environment for each node in addition to an efficient communication process based on message passing through buffer exchange, therefore minimizing interaction among nodes. The distributed architecture of Cunetsim is a fundamental cornerstone in the support of heterogeneous computing architecture. That is why the master-worker model is extended to the three-level coordinator-master-worker (CMW) model where, at the top level, the coordinator ensures the time synchronization and the load balancing among the masters (second level), each of which locally manages the time synchronization and event scheduling among the workers. At the third level, the workers execute threads performing tasks. From the coordinator point of view, the master manages multiple simulation instances, which is why the master-worker subsystem is referred to as extended logical process (ELP). Since Cunetsim aims to maximize the simulation parallelism at the event level, it introduces the notion of worker set (WS) which regroups workers that perform the same process in the same context. Based on the passive parallelism concept, events are generated simultaneously for each WS and can be processed in parallel. In the same spirit, the event scheduling policy handles two type of parallel events: (i) cloned independent events(CIE), where events differ only in data, and (ii) independent foreign events (IFE), where events differ in both algorithm and data. From a scheduling standpoint, each CIE clone is considered as a unique event to which we attach parallelization parameters that are later interpreted by the API in order to distribute those events among workers. Finally synchronization and communication processes handle domain-specific operation between CM and MW. Therefore, each master synchronizes its relative workers on one side and synchronizes its own clock with the coordinator's one on the other side. The communication between workers of the same ELP is done by a direct message exchange while workers form different ELPs need the contribution of masters of each side.

4. SCENARIO AND SETUP

In this work we propose an experimentation scenario which aims to prove the efficiency and the scalability of the cunetsim concept in terms of traffic management across the network; Moreover, we try to highlight the efficiency of the event scheduling approach and its ability to handle an uninterrupted event flow of a Ge/s event rate. For this reason, we customize the benchmark methodology proposed on [2] by defining a static network topology composed of three independent activity areas (AA) each of which follows the grid configuration where the edge of an AA contains 750 nodes as illustrated in Figure 1; thus each AA includes 562.5K nodes.[1]. The scenario includes one traffic source which generates 600 uniform 128-byte packets with 1 second of inter-departure time. All nodes forward unseen packets after a one-second delay to model the network latency whilst mediums reliability is reflected using dropping probability. Depending on the latter, each node decides whether or not to relay a received packet. The drop probability is the parameter which allows us to tune the traffic load while the density of process events remains stable. Therefore, all the experiments have been conducted while varying this probability from 0 (no rejected packets) to 1 (all packets rejected) with a 0.1 step. The used frameworks are CUDA 5.0 and OpenMPI 1.4.1. The hardware platform is constituted by one PC which includes an INTEL i7 3930k CPU (6 cores with hyper threading), 32 GB of DDR3 and three GeForce GTX860 2GB (1536 cores for GPGPU computing. The OS is Ubuntu Linux 11.10, the PGI compiler version is the 12.9 and the Nvidia driver version is 295.41.

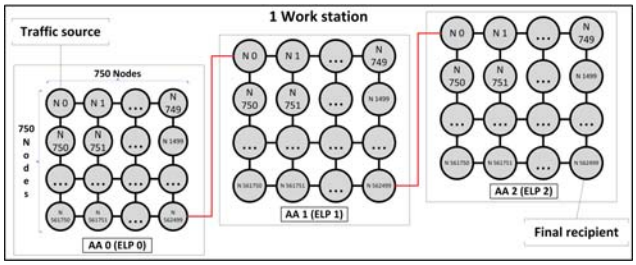
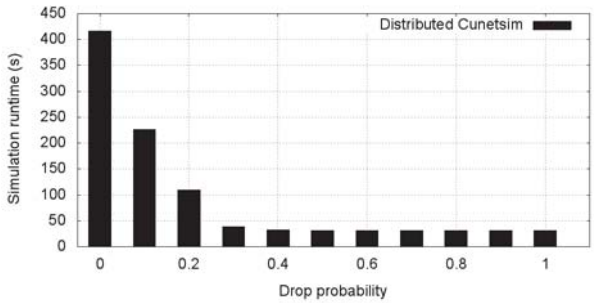


Figure 1: Topology of the benchmarking scenario

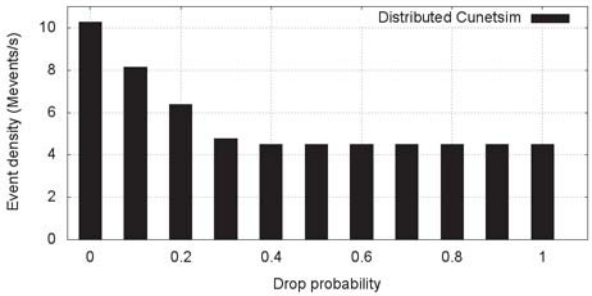
5. EVALUATION RESULTS

we propose to consider the runtime as global efficiency metric (Figure 2(a)) and the events behavior as a scalability descriptor. Thus we first study the event density defined as the average number of generated events per simulated second(Figure 2(b)); secondly we analyze the event rate defined as the average number of events processed per second (Figure 2(c)) and third we highlight the difference between the processing cost of a message and a state(Figure 2(d)). Conceptually, for a DP of 1 the simulation represents a vacuum network since there is no exchanged message which reflects the minimal simulation load where all generated events are a state's processing. The total number of events is about 4.5 billions. On the other hand, a DP of 0 means that we simulate a fully loaded network where all links are reliable. Therefore the total number of generated events is close to 10.3 billion. We observe that when the DP value is between 1 and 0.4 the number of messages remains small and the runtime did not exceed 31 seconds with a spectacular event rate of 800 million/s. However when the DP increases further the network becomes progressively loaded

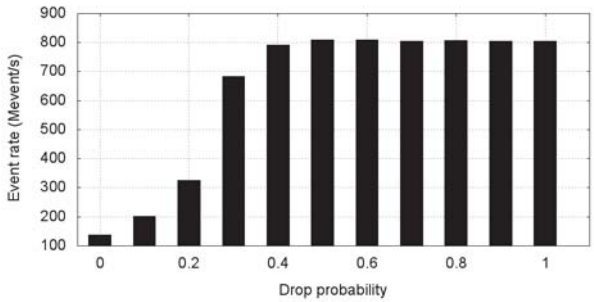
[1]That value represents the hardware limitation of the used GPU in term of memory space since each node needs 3.8 Ko



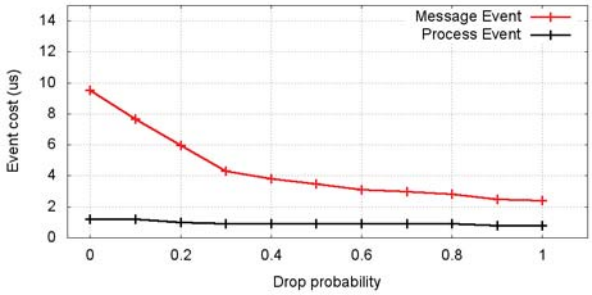
(a) The average runtime increases linearly function of the number of processed packets



(b) The event density increases when we decreases the DP



(c) Processing uniquely states allows high events rate



(d) A message cost 10x more than a state in average

Figure 2: Hardware usage rate of both CPU and GPU function of the network size

and the number of message increases rapidly. Thus the simulation runtime rises significantly. We observe however, that the events rate falls to 100 million/s which means that the framework is more sensitive to message events rather than state events. Hence, Figure 2(d) illustrates the evolution of the average time needed to process both events and we notice that a message costs 10x more; that fact is due to three reasons: first a message requests the collaboration of two workers: the sender and the receiver, secondly it required at least two memory operations one read and one write and finally it uses atomic operation to write on the destination buffer which produces a *stunts expectation* when all nodes are active. This is one of the reasons why the message cost increases when the number of messages increases.

To conclude, we can assert that Cunetsim framework provides significant results in term of runtime and ensure major scalability gain on the size of each simulated partition when considering distributed simulation.

6. CONCLUSION

In network research area, simulating large scale networks is always challenging since emerging technologies such as LTE and WiMAX allow higher and higher communication capabilities. While distributed and parallel frameworks were generally used to cope with such simulations, they did not support heterogeneous computing hardware and are mainly CPU-oriented. However, the simulation over GPU has been recently studied as an efficient solution to realize large scale experimentation. In particular, Cunetsim is a distributed GPU-based framework which targets wireless mobile networks and uses both the CUDA API for GPU integration and MPI API for the distribution over several machines and/or GPUs.

In this work we show the results of an extra-large benchmarking scenario which involves 1.5 millions nodes exchanging about 4 billions messages during 5602 simulated seconds. The simulation is done using one workstation which includes 3 GPUs each of which includes 1536 cores. The total runtime for the worst case was about 410 s which is 13x faster than the realtime execution. We note however that the size of the available memory on the GPU was the main limitation that hinder our scalability.

7. REFERENCES

[1] B. Bilel and N. Navid. Cunetsim: A gpu based simulation testbed for large scale mobile networks. In *Communications and Information Technology (ICCIT), 2012 International Conference on*, pages 374–378. IEEE, 2012.

[2] M. S. M. B. Bilel Ben Romdhanne, Navid Nikaein. Hybrid cpu-gpu distributed framework for large scale mobile networks simulation. In *16th IEEE/ACM DS-RT :The International Symposium on Distributed Simulation and Real Time Applications*. Ieee/ACM, 2012.

[3] K. Renard, C. Peri, and J. Clarke. A performance and scalability evaluation of the ns-3 distributed scheduler. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, pages 378–382. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012.