NS-3 Performance Analysis and Development of Effective Load Balancing Algorithms

Olga Lesnova, Eugene Kalishenko
Open Source and Linux lab
Computer science dept.
Saint Petersburg Electrotechnical University “LETI”
Saint Petersburg, Russia
{lesnovaolga, ydginster}@gmail.com

Abstract

The NS-3 simulator is a discrete-event network simulator. It is used both for research and educational purposes[1]. The role of performance in network simulation is very important now, because modern networks become larger and their complexity is increasing. The goal of this work is to analyze existing implementations of parallel algorithms (distributed version[2] and multithreaded version[3]) and to develop a new effective load balancing algorithm or improve the current implementations.

Index Terms: NS-3, Load balancing, Performance analysis.

I. Introduction

The NS-3 project was started as an open-source in 2006. It has a flexible object-oriented and plug-in architecture for creating different network topologies and extend it with new traces output formats[4]. NS-3 can work with different kind of networks from small LAN to huge ad-hoc or WMN (Wireless mesh networks). Large networks simulation may take much time: several hours and more, that's why performance analysis and optimal load balancing is so important. Often there is a need to vary some parameters and get statistics several time on the same model, so the urgency of simulation performance improvement is obvious. High-performance computing is the modern field in computer science. Nowadays most PCs support different concurrent technologies and modern software should be developed to achieve maximum performance on every SMP platform.

II. Main part

A. Distributed version

Starting from version 3.8 (2010) NS-3 includes support of distributed simulation mode, based on MPI standard. Distributed simulator is implements “Conservative lookahead” algorithm. Simulation divided into slightly coupled logical processes, which can be placed on a different processors for execution [2].

At first the value of lookahead as the minimal delay from all pairs of nodes from different logical process is computed. Then simulation goes gradually increasing next synchronization time, alternating independent simulation in logical processes and synchronization step (Fig. 1).

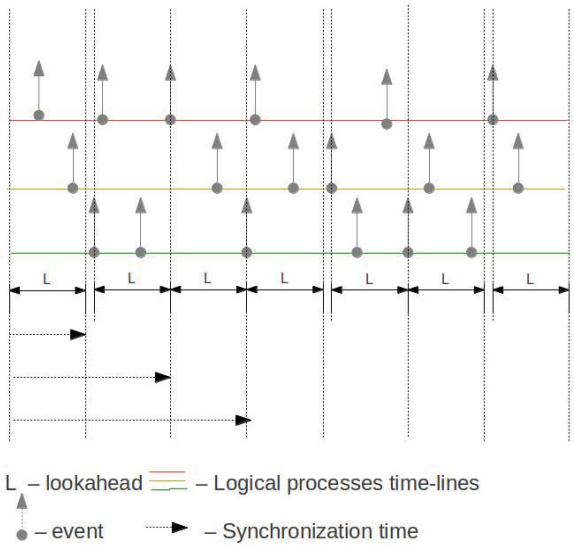

Fig. 1. Distributed algorithm example

The main disadvantage of this implementation is manual nodes distribution among processes. Also, MPI technology requires additional hardware. The other way to increase performance is to make implementation multithreaded.

B. Multithreaded version

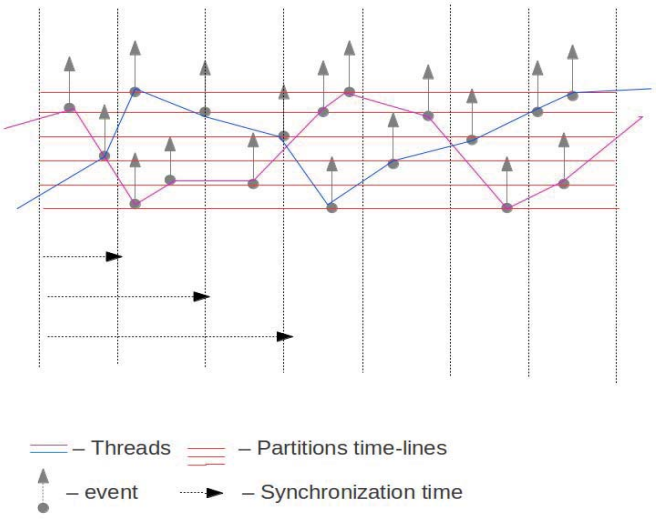

Fig. 2. Multithreaded algorithm example

In 2009 there was a project aimed to implement parallelism in NS-3, based on the “Barrier-based algorithm” for synchronization during simulation process. All nodes considered as different partitions (one node per partition) distributed among threads automatically. Lookahead value is different for each partitions and means the minimal delay between partition and all neighbors. Every step of algorithm one thread computes global max time, before which simulation in each partition can be done independently. Then all threads take partitions one by one and execute all events from corresponding queue before the given time (Fig. 2).

C. Performance analysis

Performance analysis results are presented on Figure 3. The DARPA NMS Campus Network model was used for tests. For performance measurement of default and multithreaded versions the following configuration was used: Intel(R) Core(TM) i5-2320 CPU @ 3.00GHz X 4 processor. For distributed version cluster consists of 2 nodes with the same characteristics.

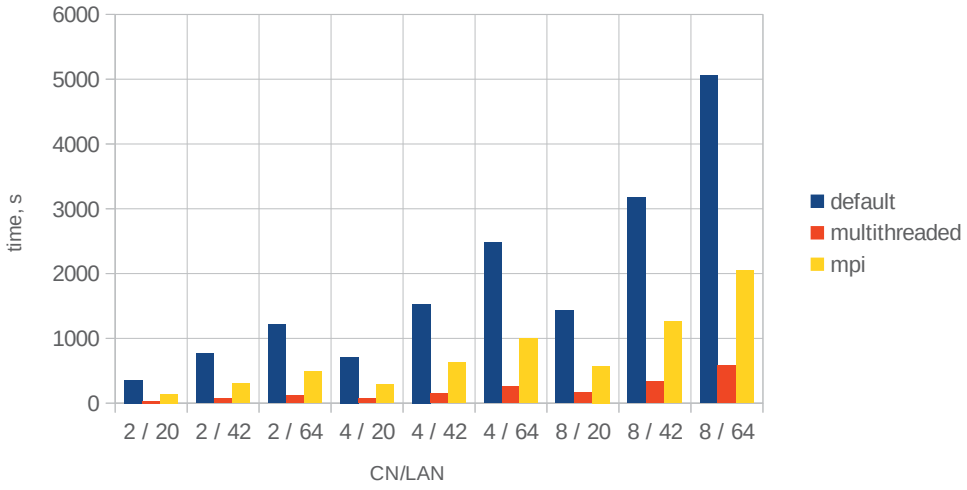

Fig. 3. Time measures for run time.

On Figure 3 one can see that asymptotically run time in all versions grows the same way. Multithreaded version works better then others, but now there are some doubts about that fact and it will be analyzed deeper. In order to make more precise conclusions deep performance analysis should be carried out.

D. Default version profiling

Default version was analyzed with Valgrind and Intel Parallel Studio.

The Top Hotspots from parallel studio profiling:

1) ns3::MapScheduler::Insert 19.527s
2) ns3::SimpleRefCount<ns3::Object, ns3::ObjectBase, ns3::ObjectDeleter>::Ref 14.854s
3) ns3::Ipv4Interface::GetNAddresses 8.925s

4) ns3::operator<(const Scheduler::EventKey &a, const Scheduler::EventKey &b) 8.812s
5) ns3::SimpleRefCount<ns3::Object, ns3::ObjectBase, ns3::ObjectDeleter>::Unref 7.480s
[Others] 468.521s

Valgrind results are not belie parallel studio results. Two obvious ways to increase performance are: more effective map implementation and smarter reference counter. But this improvements are not so effective as using concurrent algorithms, which can give more advantages.

III. Conclusion

Current research directions:

1) Profile the existing single and multithreading implementation in order to find main hotspots, cache misses etc.
2) Develop an effective load balancing algorithm for network nodes redistribution among the cluster nodes before and during the simulation.
3) Try to integrate multithreading in current NS-3 version.
4) Optimize multithreading implementation using profiling results.

References

[1] Selim Ciraci, Bora Akyol “An Evaluation of the Network Simulators in Large-Scale Distributed Simulations” Pacific Northwest National Laboratory Richland, WA, USA, http://gridoptics.pnnl.gov/images/e/ee/Evaluation_of_the_Network_Simulators.pdf.
[2] Joshua Pelkey, George Riley “Department of Electrical and Computer Engineering ” Georgia Institute of Technology, Atlanta, GA. USA, http://users.ece.gatech.edu/~riley/ece6110/handouts/DistNS3.pdf.
[3] Guillaume Seguin “Multi-core parallelism for ns-3 simulator ”, Internship at INRIA Sophia-Antipolis, PLANETE team , 2009, http://guillaume.segu.in/papers/ns3-multithreading.pdf.
[4] NS-3 description, http://www.nsnam.org/wiki.