Contribution ID: **12** Type: **not specified**

Computationally Modeling High-Speed Scientific Networks

Tuesday, 11 September 2012 14:22 (12 minutes)

Computationally Modeling High-Speed Scientific Networks
Jun Yi, Venkatram Vishwanath, and Rajkumar Kettimuthu
Mathematics and Computer Science Division, Argonne National Lab
{jyi, venkatv, kettimut}@mcs.anl.gov

1, The Need of Computational Modeling for Scientific Networks Scientific experiments (e.g., climate modeling and prediction, biomedical imaging, geosciences, and high-energy physics) are expected to generate, analyze, and distribute data volumes on the order of petabytes. These experiments are critically dependent upon advanced high-speed networks to move their enormous data between local and remote computing and storage facilities. These experiments usually have a wide range of networking requirements and characteristics, e.g., bulk data transfer (high bandwidth, loss-less), climate visualization (large bandwidth, less jitter), and real-time data analysis and decision making (high bandwidth, low latency, and loss-less). Moreover, these scientific data flows usually traverse multiple different networks (shared WAN, dedicated circuit-based WAN, and LAN) and are transferred using different protocols for better performance (For example, GridFTP or parallel TCP over high-bandwidth
large-delay networks, UDT over high-latency networks). The heterogeneity of networks, flows, and protocols and the ever-increasing traffic volume challenge the scientific network planning, operation (e.g., troubleshooting and configuration), and design to satisfying the heterogeneous requirements of scientific experiments.

Here we present two use cases that will benefit from the computauional modelling of end-to-end large scale networks. DOE's Advanced Photon Source (APS) user facility at Argonne National Laboratory provides the Western Hemisphere's most brilliant X-ray beams for research. It is projected to generate 100+ terabyte per day within the next year. As data volumes increase, experiments may have to use external supercomputing facilities to process the data in real-time, which will necessitate additional fast WAN transfers. The Office of Science at DOE projects to support such high speed transfers as APS and comes across a challenging question naturally: how to evolve from current high-speed network. Adding hardware capacity and improving software efficiency are two solutions. However, where to add these hardware capacity and how/where to improve software efficiency cost-effectively need scientific, not intuitive or hypothetical, answers. A computational model of scientific networks can answer the question precisely: we can experiment on various network configurations with the data communications requirements that the Office of Science at DOE collects yearly from scientists, and choose the one that satisfies their requirements with the lowest cost or lowest evolution effort.

Globus Online and Globus GridFTP provide high-throughput, reliable, and secure big data movement service, but the throughput in most cases, is still far from the physical network capacity. To further improve the throughput of a GridFTP transfer, an end-to-end approach can help to identify the bottleneck and optimally choose routing and transportation protocols and parameters along the path, which, however, can not be fully effective until we have deeper understanding of interactions among various data flows and between the flows and network. With a computational model, the network configuration based on current status can be modeled and experimented to locate performance bottlenecks and choose the optimal configurations from the vast configuration space in real-time.

2, The Challenges of Computational Modeling for Scientific Networks
However, it is challenging to computationally model scientific networks, even with the massive computing, networking, and storage capacities provided by today's supercomputers, grids, and clouds. These challenges originate from the demands of computational models:
[1] Scalability. The model should be able to represent networks of various scales. The execution of the model should fully harness the resources of future exascale computing facilities.
[2] Accuracy. The model should be as accurate to the real world as possible. It should accurately predict performance results (e.g., throughput, latency, transfer completion time, resource usage) under various traffic uncertainties with appropriate levels of computation loads within a certain time limit.
[3] Composibility/Extensibility. Models of two interconnecting regional
networks should be able to easily compose to form a larger network without knowing the internal details of each model.

Among all of these challenges, the scalability challenge is the most critical one from our perspective. If a network model can not take full advantage of the underlying massive parallel computing capacities, it can not

produce accurate results within a certain limited time, not even to mention experiments on a larger composed networks.

Existing network modeling techniques will not be effective considering the sheer scale and complexity of scientific networks. The discrete packet-level event simulation method usually uses a central discrete event scheduler to schedule the time-stamped events such as packet and timer expiration. This centralized scheduler becomes both the communication and computation bottleneck at scale. For example, 1M events per network processing entity (e.g., network reception interrupt handler) per second and 1M vertical or horizontal network processing entity for a network of medium scale will generate 1T events to be delivered, processed, synchronized, and persisted in a second. Assuming 256 bytes for each event packet, 256T bytes per second traffic will be generated, which will overwhelm current and forthcoming aggregated network capacities of supercomputers and grids (several terabytes per second). Moreover, event handlers must be executed in ascending timestamp order to preserve the semantics of the physical network. Further, an executing event handler may generate new events, which should be queued temporally as well. This method is not scalable computationally due to the implicit fact that events are processed serially.

The pivot issue that prevents network models from harnessing the massive computing/communication capacity lies mainly in the process/event synchronization method. On the one hand, massive amount of data and control information flows among network elements and any patterns of dependency may exist and change dynamically (e.g., a single event may have a ripple effect over the entire network). The complicated interdependency naturally requires a serial execution of events, which exacerbates the modeling performance when combined with large scale network models. For example, the tardiness of a single element may extremely waste computing resource as a whole (e.g., many entities wait for the completion of a tardy entity at a certain lock step) and slow down the entire model. On the other hand, the capacity of today's supercomputers, grids, and clouds can only be fully harnessed by programs exhibiting massive parallelisms. The execution of a serial program makes no much difference on a desktop computer or supercomputer.

Existing event/process synchronization methods in the distributed system literature is not efficient at scale. Most existing synchronization methods only deals with logical (not timeliness) dependency. Just until recently, parallel discrete event/process simulation was merely an academic research topic. There are basically two methods to impose the correct temporal order of distributed event execution: conservative and optimistic methods. By a conservative method, only a safe event can be executed (if a process contains an event E1 with timestamp T1 and the process can determine that is impossible to receive an event with a smaller timestamp, then the process consider that executing event E1 is safe without violating temporal constraints). Unsafe events must be blocked. Consequently, most events processings are blocked in most time. By an optimistic method, the temporal relationship between events can be broken but a detection and recovery mechanism is added: whenever the incorrect temporal order of events is detected a rollback mechanism is invoked to recover. However, deadline deadlocks are frequently formed and hard to detect in a complicated and large network and therefore large computation is spent on expensive deadlock detection and recovery. Moreover, the running pace of events in the model rarely match those of the physical network and the violation of temporal relationships exists everywhere (due to pervasive and complicated dependency among network elements) and thus most computation is spent on computation rollbacks.

3, Our Basic Idea and Its Challenges

We propose a hierarchical method for modeling scientific networks. The basic idea is to organize temporal relationships, event delivery, and work flow execution hierarchically to reduce communication overhead, increase the parallelism of event processing, and dynamically balance event processing workload in a synchronization-aware manner. The entire model is organized as a tree with (either vertical or horizontal) network elements as leaves. Each interior node comprises a temporal synchronizer (TS), a workload distributor (WS), and an event forwarder (EF). For example, multiple network processing elements (e.g., routing, queue, and transport elements) of a networked computer can serve as leaves under the same parent. Multiple TSs, WSs, and EFs corresponding to multiple interconnecting physical network elements can serve as children of a common interior node corresponding to an autonomous system.

The hierarchy allows disjoint subtrees to run in parallel provided that the difference of their simulation times is no greater than the communication latency between them. A TS is responsible for synchronizing the event processing within its subtree. It uses the existing tree-barrier method to solve the communication bottleneck problem, e.g., it reduces temporal communication load and latency since a few messages cross the hierarchy can progress the entire model to next time step. Moreover, it avoids time-consuming and complicated event synchronization deadlock detection and recovery by the conservative and the optimistic synchronization methods.

An EF is responsible for forwarding data and control events between interconnect (either vertical or horizontal) network elements, which further reduces communication workload since absolutely majority of control and data flows within the same subtree. Moreover, since EFs forward all timestamped events, they are able to accurately estimate the needed resource by any subtree in any specific time range, which facilitates distributing workload to the underlying distributed and parallel execution environment. A WS is responsible for

distributing event processing workload within its subtree (e.g., to accelerate event processing if it lags behind other subtrees). We do not use optimistic event synchronization methods to seek opportunistic event independence at the cost of expensive computation rollbacks, which we believe can only accelerate turnaround time of a model of scientific networks to a limited extent. Alternatively, we resort to synchronization-aware, hierarchical, and dynamic load balance approach, where sluggish subtrees will be timely allocated sufficient resource to keep pace with the remainder of the model.

However, we still face many challenges to pursue this approach. We list a few of them:
[1] Configurations. How to decide the size, height, and descendents of subtrees is important to the efficiency of this approach. We intend to adaptively configure the tree and subtrees according to the network element interaction pattern of the physical network and the capacity and workload of the underlying parallel and distributed execution environment.
[2] Workload distribution. Distributing the event processing within a subtree appropriately into concrete execution entities (e.g., threads within the same process, threads/processes distributed across multiple cores or heterogeneous machines, etc.) greatly affects the running time of the model due to changing communication latency and workload distributions in each simulation time-step. A dynamic workload- and synchronization-aware scheduling framework is needed.
[3] Fault tolerance. TSs, EFs, and WSs will be single points of failure and redundancy mechanisms are needed to recover from failures. We intended to use existing replication technique to increase the robustness of this approach.
[4] Group communication. TSs need to broadcast timestamps within theirs respective subtrees. Effective group communication mechanisms will extremely improve the efficiency of this approach and therefore is worth investigating.

Primary authors: Dr KETTIMUTHU, Rajkumar (Mathematics and Computer Science Division, Argonne National Lab); Dr VISHWANATH, Venkatram (Mathematics and Computer Science Division, Argonne National Lab)

Presenters: Dr KETTIMUTHU, Rajkumar (Mathematics and Computer Science Division, Argonne National Lab); Dr VISHWANATH, Venkatram (Mathematics and Computer Science Division, Argonne National Lab)