Load Distribution in Large Scale Network Monitoring Infrastructures

Josep Sanjuàs-Cuxart, Pere Barlet-Ros, Gianluca Iannaccone, and Josep Solé-Pareta

Universitat Politècnica de Catalunya (UPC)
{jsanjuas,pbarlet,pareta}@ac.upc.edu
Intel Research Berkeley
gianluca.iannaccone@intel.com

Abstract. We aim to build a scalable, distributed passive network monitoring system that can run several arbitrary monitoring applications on a number of network viewpoints. We explore the research challenges associated to the resource management of such a system, and propose a generic architecture that enables the computations of monitoring applications to be spread across a set of nodes.

1 Introduction

Network monitoring is becoming a necessity for network operators, who usually deploy several monitoring applications that aid in tasks such as traffic engineering, capacity planning and the detection of attacks or other anomalies. General-purpose network monitoring systems are emerging to support multiple concurrent applications that extract useful data by examining the packets that traverse a network link [1–3]. There is also an increasing interest in large-scale network monitoring infrastructures that can run multiple applications in several network viewpoints [4].

Network monitoring infrastructures are extremely resource constrained, due to the continuous growth of network link speeds and computational complexity of monitoring applications. It is therefore highly desirable for such systems to be scalable (e.g., to incrementally add more computing nodes to the system) in order to support more applications and to sustain a higher volume of traffic. However, providing network monitoring applications with the ability to migrate across nodes is not trivial. The solution of simply replicating the incoming traffic to other nodes is prohibitively expensive given the bandwidth requirements of replicating captured traffic across the infrastructure.

Furthermore, such infrastructures must efficiently deal with hot spots that monitoring applications naturally create, since the events of interest are usually localized (e.g., intrusion and anomaly detection). Such events can cause load to be distributed unevenly across the monitoring infrastructure.

In this work, we review the principal research challenges behind building a distributed network monitoring system to support the execution of several monitoring applications in multiple network viewpoints. The challenges in network monitoring systems differ from the ones traditionally explored in distributed systems research [5] in that monitoring applications are continuous and never finish (as opposed to jobs with known computation time and deadline). This different workload, as we will see in the next sections, severely restricts our design options.

2 Research challenges

System distribution. A natural way to provide systems with scalability is to distribute its load across many nodes. The administrators of the system can then increase the resource pool of the system easily by providing additional computing nodes to the system. However, distributing a network monitoring system is not easy, due to the high data rates associated to network traffic.

Creating replicas of the traffic for simultaneous processing in different nodes is undesirable, since it involves provisioning the network monitoring system with several times the amount of bandwidth of the monitored network link. Additionally, it would be more expensive, since it would involve additional traffic replication and collection devices. Another option to provide scalability would be to split the traffic of each capture point across several co-located nodes, or else, connected by a high speed network to the capture point. This approach would require network monitoring applications to be implemented with distributed algorithms, since they would be running simultaneously on all of the nodes, thus raising the implementation complexity. However, a very important characteristic of a successful network monitoring platform is that applications can be specified easily by end-users [1, 2].

We propose a generic architecture for a distributed network monitoring system. The system is comprised of two kinds of logical nodes: capture nodes and analysis nodes. The only difference between these nodes is that only capture nodes receive network traffic, whereas analysis nodes serve only to increase the computing resources of the system as a whole. Capture and analysis nodes represent functional components. They can be co-located on the same physical system. The operators of the infrastructure can easily increase the resource pool by adding extra systems that only run analysis nodes. Only adding extra capture nodes involves a carefully planned action, since it will require placing the required traffic capture equipment.

However, in network monitoring, not all tasks are suitable to be run in said analysis nodes, since replicating the traffic is not an option. We propose a two stage architecture for network monitoring applications that addresses this problem. The first stage of each application performs a first data filtering and aggregation step, with data volume reduction in mind, and requiring as light computation complexity as possible. The results of the first stage are received by the second stage of the application, which computes the final results without requiring access to the raw network traffic stream. Tasks such as CPU intensive computations or those requiring long term state are suitable to run in the second stage. The complete architecture of the system is presented in Figure 1.

The advantage of structuring network monitoring applications this way is that, while the first stage of each application is pinned to its corresponding capture point, the second stage can be migrated to any other system (be it a

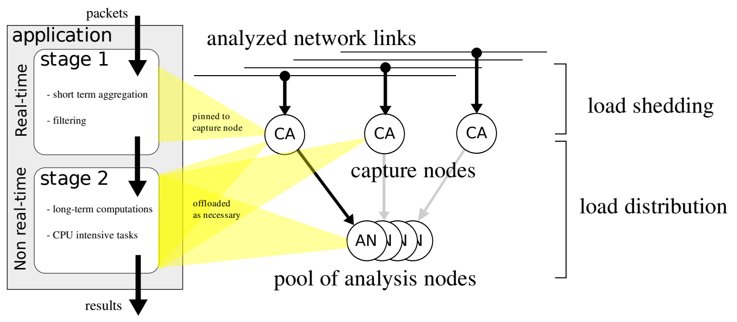

Fig. 1. Two stage application and overall system architecture.

system running a capture node but with sufficient idle resources, or a system dedicated to analysis nodes), due to the data volume reduction, which enables sending the intermediate results from the two stages across nodes.

This scheme has important additional advantages. First, it isolates the tasks that require access to the raw network traffic stream. Because of the high data rates involved, buffering network traffic can only be done for short time spans. A second important benefit of this architecture is that it encourages light computations on capture nodes, which are the most resource constrained points of the system.

Load shedding. Capture points cannot be easily scaled, as already explained. In the event that a capture point cannot cope with the resource demands of the first stages of each application to process incoming traffic, part of the computations have to be discarded in order to avoid uncontrolled packet loss. Usually, the solution to this problem involves sampling the incoming traffic. In the recent past, we have been exploring the problem of dynamically selecting the most appropriate traffic sampling rate in a network monitoring system comprised of only one capture point. The results of this research [6, 7] can be applied to each of the capture nodes of the distributed network monitoring system.

Distributed scheduling. Once the system is provisioned with the capability of distributing its load, it is critical to find a proper assignment of tasks to nodes. While the aggregate capacity of the system can be enough to cope with the load imposed by network monitoring applications, incorrectly assigning tasks to nodes can lead to (*i*) excessive traffic sampling by the load shedding subsystem, when second stages are run on an already overloaded node, and (*ii*) unnecessary delay in computing the final results of network monitoring applications, in the event that a load imbalance causes overload in some of the nodes while others are idle.

A proper assignment of tasks should have the following properties: first, the second stages of network monitoring applications should not run on any capture node that is sampling the incoming traffic, as long as there is sufficient spare resources in the analysis nodes. Second, the load across the analysis nodes should be balanced in such a way that the maximum delay in computing the final results of each application is minimized.

In an infrastructure comprised of n systems and m network monitoring applications, n^m assignments of second analysis stages to systems are possible. Clearly, the scheduling mechanism of the system will not be able to explore all the possibilities. It will be important to cut the search space with efficient heuristics in order to find desirable task allocations. When the system capacity is not sufficient, as in the case of capture nodes, the system will need to resort to load shedding.

Also, it is important to note that, due to the continuous nature of network monitoring applications, the scheduler cannot rely on simply providing an initial assignment. Since the load of applications will depend on the algorithms they implement and the incoming network traffic, the assignment must change over time. Therefore, dynamic reassignment through migration will be necessary in order to rebalance the load across the nodes.

We are currently developing a simple scheduling algorithm based on the idea of pairwise node load exchanges. Periodically, the nodes of the system pair randomly with another node Then, both nodes estimate their future workloads, and attempt to exchange tasks in a way that they balance their workloads.

Migration costs. As already noted, task migration is the key mechanism to achieve a good distribution of the load across the system's nodes. Our architecture provides the following migration procedure: 1) the application is paused, 2) it is asked to serialize its state, 3) its code, state and queued input are transferred to the destination node, 4) it is loaded and asked to deserialize its state, and 5) it is resumed. Migrations introduce a series of overheads to the system. First, applications' execution is delayed by stopping and transferring their state over the network. It is interesting to note that, during its execution, the size of an application's state may vary heavily. It is important that the scheduler chooses the right time to migrate an application. Second, in the event that the application is being offloaded from a capture node, the results from the first stage have to be continuously sent to the receiving node. Therefore, the scheduler has also to consider bandwidth constraints before migrating an application.

Indexing the results. Due to the distributed nature of the system, the second stage of each application can run on different nodes, spreading its results over multiple locations. Therefore, the system requires an indexing and result retrieval mechanism (e.g., [8]).

3 Use Case Scenario

In this section, we present a simple use case that validates the benefits of our distributed network monitoring architecture. We use the CoMo system [2] to

Application	*Load Shedding*	*Load Distribution*	*Gain*
autofocus	89.2%	100%	12.1%
pattern search	25.8%	47.7%	84.9%

Table 1. Application accuracy on a centralized vs. a distributed setting.

implement our architecture, since it is a flexible platform that supports user-defined network monitoring applications, and runs realtime and non-realtime tasks separately. For the sake of reproducibility, in our experiments we use a 30 minutes long network traffic trace collected at the access link of the Technical University of Catalonia (UPC), which connects 10 campuses, 25 faculties and 40 departments to the Internet through the Spanish Research and Education network (RedIRIS).

We implement two different network monitoring applications. First, an IDS-like pattern search application, that tries to match 10 patterns to the input traffic. Second, Autofocus [9], a tool that aggregates traffic in clusters and reports the ones that generate a large volume of traffic. We use this application to aggregate the traffic at different time intervals of 1, 30, 60, 120, 300, 600 and 1800 seconds. These aggregation intervals are somewhat arbitrary, however they are set to generate a variable application workload that emulates a real system configuration.

Pattern search is an example of a network monitoring application that cannot scale easily, since all computations require direct access to the packet stream, i.e., have to run in the capture node. Conversely, Autofocus perfectly suits our architecture. In the first stage, it records all the 5-tuples of the packets, while the second stage performs the aggregation at the different time scales.

First, we run the complete system on a single processor. We observe that, with both applications, the 30 minute long trace is processed in 77.17 minutes, indicating that the system is overloaded. Next, we enable the load shedding subsystem, which discards the necessary amount of load in order to process the trace in real time (i.e., in 30 minutes). The system selects a sampling rate for each application that corresponds to the fair share of the available CPU time, using the scheduling algorithm described in [7]. Since both applications' CPU demand exceed the available resources, and their resource demands are similar, it applies a similar sampling rate to both, that averages to 25.9%.

We finally offload the second stage of Autofocus to a separate processor that is running an analysis node. This has the effect of freeing CPU time in the capture node. This effect is magnified by the fact that, in Autofocus, the bulk of the cost is in the second stage. As a result, the sampling rate dramatically improves. Autofocus does not exceed its fair share of the CPU, and therefore its sampling rate is 1, and can now compute an exact result. In turn, pattern search can use the cycles that Autofocus has offloaded, and now operates with an average 47.6% sampling rate, thus processing almost twice the amount of traffic.

Table 1 compares the accuracy of each application's results, which is computed as in [7].

4 Conclusions

We have presented the research challenges associated to building a system suitable for large scale network monitoring that supports multiple concurrent monitoring applications over several network viewpoints. We have proposed a two stage architecture for network monitoring applications that separates them in two stages, one that has access to the raw packet stream and generates intermediate results, and a second one that calculates the final results and includes CPU-intensive tasks and long term state maintenance.

The principal advantage of this architecture is that a distributed network monitoring system comprised of several nodes can migrate the second stages away from the original capture node. Therefore, the system is free to rearrange the applications in the most convenient way, thus alleviating the hot spots in the infrastructure. We have illustrated the advantages of our architecture with a simple use case scenario. We were able to significantly increase the sampling rate of the system by offloading computations from the capture node.

Our future work is centered on building the distributed scheduler of the system. Since the workload of applications is highly dynamic and traffic dependent, the scheduler has to monitor the state of the system and constantly evaluate the need for application migration. Also, since the cost of migration is not negligible, the scheduler must ensure that it will amortize the costs of migrations while aiming to balance the load of the system, considering the bandwidth constraints of the infrastructure.

References

1. Cranor, C., Johnson, T., Spataschek, O., Shkapenyuk, V.: Gigascope: A stream database for network applications. In: Proc. of ACM SIGMOD. (June 2003)
2. Iannaccone, G.: Fast prototyping of network data mining applications. In: Proc. of Passive and Active Measurement Conf. (PAM). (March 2006)
3. Reiss, F., Hellerstein, J.M.: Declarative network monitoring with an underprovisioned query processor. In: Proc. of ICDE. (April 2006)
4. Claffy, K.C., et al.: Community-oriented network measurement infrastructure (CONMI) workshop report. ACM SIGCOMM CCR **36**(2) (April 2006)
5. Casavant, T.L., Kuhl, J.G.: A taxonomy of scheduling in general-purpose distributed computing systems. IEEE Trans. Softw. Eng. **14**(2) (February 1988)
6. Barlet-Ros, P., et al.: Load shedding in network monitoring applications. In: Proc. of USENIX Annual Technical Conf. (June 2007)
7. Barlet-Ros, P., et al.: Robust network monitoring in the presence of non-cooperative traffic queries. Computer Networks (in press)
8. Li, X., et al.: MIND: A distributed multi-dimensional indexing system for network diagnosis. Proc. of IEEE INFOCOM (April 2006)
9. Estan, C., Savage, S., Varghese, G.: Automatically inferring patterns of resource consumption in network traffic. In: Proc. of ACM SIGCOMM. (August 2003)