Enabling Online Visualization Through Distributed Trace Collection

Rafael Keller Tesser, Lucas Mello Schnorr, Philippe O. A. Navaux
Instituto de Informática
Universidade Federal do Rio Grande do Sul
CEP 91501-970 - Porto Alegre - RS - Brazil
E-mail: {rktesser, lmschnorr,navaux}@inf.ufrgs.br

Abstract

The monitoring of distributed systems helps to understand the behavior of programs and resources. One way to do this is through the use of visualization techniques. In order to generate this visualization, one needs a means to gather the data which is scattered throughout the system. The work presented in this paper uses a hierarchical distributed collection model to perform this task. This model is composed of three kinds of components: collectors, aggregators and clients. The collectors get the monitoring data generated locally in the resources and send them to a set of aggregators. These forward the data to higher level aggregators or to clients, forming an hierarchical structure. The clients are responsible for providing the received data to an object which uses them to do some kind of processing and analysis. In this work, this role is played by the DIMVisual model, which integrates the data and generates a visualization. The collection model also includes a subscription mechanism which allows the clients to select from which collectors to receive data. This reduces the transmission of unnecessary data. In order to achieve the online visualization of the data, the collection model can be used together with the DIMVisual Pajé Reader, whose development is also detailed in this paper.

1. Introduction

Monitoring systems have been used to help the understanding of applications and distributed resources. They can benefit other areas such as task scheduling, performance analysis and resource failure detection. Grid platforms can also benefit from a monitoring system by allowing application developers to observe resource utilization during the execution of parallel applications. Some characteristics of Grids that change the modeling and implementation of monitoring systems include dynamism, heterogeneity and geographical distribution of the resources.

The monitoring data is also scattered, in the case of Grids and large-scale distributed systems. Because of this, monitoring systems must have a scalable collection mechanism to centralize the information. This is needed in order to be able to correlate information from different sources. A common approach to model these mechanisms is to use hierarchical structures. For instance, Ganglia [4] and Globus' MDS [1] use this type of structure. The former implements a pull algorithm to receive data from different domains, using polling to detect the availability of new data. Inside a domain, it uses multicast to propagate the data. The latter, in its fourth version, can use both pull and push algorithms to collect monitoring information. Between these, the push model is better for observing a set of data for a period of time [1]. The problem of these approaches is that the centralization of information is usually not directly connected to an analysis environment. This connection enables an online understanding of the information, as soon as its collection happens.

The work presented in this paper proposes a novel hierarchical distributed collection model to gather monitoring information that can be attached to a visualization and analysis tool. The model is composed of three kinds of components: collectors, aggregators and clients. The collectors get the monitoring data generated locally in the resources and send them to a set of aggregators. These forward the data to higher level aggregators or to clients, forming an hierarchical structure. The clients are responsible for providing the received data to an object which uses them to do some kind of processing and analysis. In this work, this role is played by the DIMVisual model [6], which integrates the data and generates a visualization. An initial implementation to be used in distributed systems such as the Grid'5000 platform is also presented. The paper also details the development of the DIMVisual Reader, which aims to connect the hierarchical collection model output with the Pajé visualization tool. This combination allows the online visualization of monitoring data.

The rest of this paper is organized as follows. Section 2 presents the related work. Then, in section 3, we present the proposed hierarchical collection model. Section 4 presents the architecture and behavior of the DIMVisual Pajé Reader. We end the paper with a conclusion with a summary of our achievements and a discussion about future work.

2. Related Work

Ganglia [4] and Globus' MDS [1] are well known monitoring tools which use an hierarchical structure to collect data. The former can be configured to collect data from multiple administrative domains, allowing the user to visualize all information through a single web page. Grid'5000 has a basic installation of Ganglia, configured to collect data from all sites and clusters. This tool uses multicast to send data inside an administrative domain. This way it makes every node to have all monitoring information about his domain. This technique can be seen as a push data receiving model, because when a resource generates new monitoring information, it's sent to the others without their requisition. On the other hand, Ganglia uses a pull model to aggregate data from different administrative domains, using polling to discover the availability of new data. The Globus' MDS, in its fourth version, implements the push model through the use of an web services notification mechanism. However, the main concerns of this tool are resource discovery and to send out warning messages, instead of the behavior analysis of systems or parallel applications. The hierarchical model, described in the next session, uses the push data receiving model. In addition, it is main goal is to aid the analysis of the behavior of observed resources.

3. Hierarchical Distributed Collection Model

This section presents the hierarchical and distributed collection model. It is composed of three kinds of components: collectors, aggregators, and clients. Their organization forms an hierarchical structure to gather monitoring data collected in the resources of a distributed system. The model uses the push model for data receiving. This means that the transmission of data is started by its sender when a certain condition is met, without the need of the receiver to ask for it. This is good from the point of view of a monitoring system intended to perform an analysis of the behavior of observed objects during a period of time. If the aim is to be able to determine the state of the objects in a specific moment, then a pull model would be better [1]. In addition to this, a subscription mechanism is used to reduce the sending of unnecessary data. This mechanism allows receivers to opt out of receiving certain types of monitoring information.

A collector is responsible for obtaining the data generated by a monitoring tool or by an instrumented program, and finally sending this data to a set of aggregators. An aggregator forwards the data it receives to higher level aggregators and to clients, forming the hierarchical structure. The organization of the resources of most multicluster and grid systems can be seen as an hierarchy. For example, nodes in the lower level, clusters in the second, sites in the third, and so on. Because of this, an hierarchical model is suitable for use in this kind of environment. A client is the top-level component of the hierarchy. It receives the data from a set of aggregators, providing the information in a centralized way easing its analysis. We intend to use DIMVisual to process information from various sources and generate an integrated visualization.

The subscription mechanism works by keeping track, in the collectors and aggregators, of which components have subscribed to receive data from an specific collector. In its initialization a collector sends setup data, including an unique identification and its type, to register itself in the aggregators that are connected to it. This information is sent up through the hierarchy until it reaches the top level. When a client connects an aggregator, it gets the information for every collector it can receive data from. A subscription request is propagated down in the hierarchy. This propagation stops when it reaches either a component that is already subscribed to the wanted data or the collector that generates it. A collector sends its data only to corresponding subscribers. When an aggregator receives data, it checks its own list of subscribers to the originating collector, and forwards it to them. One component only knows which are the subscribers that are directly connected to it. The final destination as well as the path traversed by the data sent by a component are both unknown to it.

One desirable feature is to be able to generate an online visualization of the behavior of the system and of the execution of programs. To enable this, the collection model can be integrated with the DIMVisual Pajé reader, which is presented in section 4.

3.1. Implementation

The hierarchical collection structure has been implemented using distributed objects (DO) of the Objective-C programming language. One class has been implemented for each component of the model: DIMVClient, DIMVCollector and DIMVAggregator. These classes implement together the construction and initialization of the hierarchy, the transmission of monitoring data and subscription tasks. They implement the part of the model which is independent of the data source and format, and therefore facilitate the construction of customized components utilizing them.

A hierarchy instantiation organized with the basic classes is illustrated in the figure 1. We can see two types of collector (CollectorType1 and CollectorType2) which make internal use of an instance of the class DIMVCollector. The collector just needs to implement the code

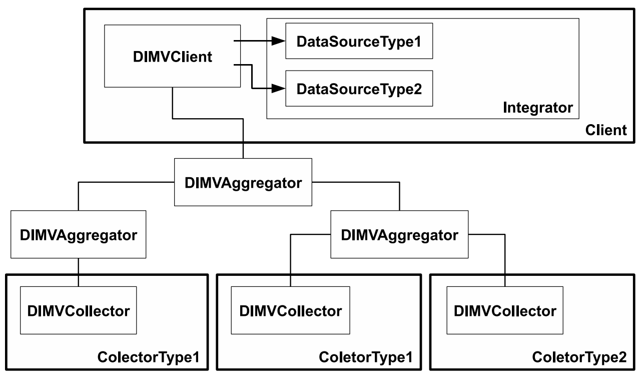

Figure 1. Components of the collection model implementation.

to collect the specific monitoring data and store it in an serializable object, that can be sent through the network using the distributed objects of Objective-C. A serializable object in Objective-C means that a given instance conforms to the NSCoding protocol. After obtaining the data, a method of the DIMVCollector object should be called, passing the data as argument. Then this object takes care of its transmission to the appropriated components.

The client instantiates an object of the DIMVClient class, specifying things like to which aggregators connect itself. Then it can pass a set of references to objects called data sources to it, specifying a type of collector for each of them. When the DIMVClient object receives data, it checks the type of the collector which generated the data. Then it calls the method `getData`, from the data source which is associated to it, passing the data as argument. In the example shown in figure 1, the data sources are objects from a larger component called integrator, such as the original DIMVisual model implementation.

The DIMVAggregator class does everything an aggregator must do, as extensively stated in previous subsection. The only aspect its user really needs to implement is its initialization, passing the appropriated arguments. The objects of the three classes must be initialized with an unique identification and a list of aggregators which they should connect to. Additionally, the DIMVCollector needs to receive the type of the collector as an argument to its initialization method.

The implementation of the components of the hierarchical collection model is mostly complete. Together, they can be used to create hierarchical structures that collect different types of monitoring data. The main classes, for example, can be extended to receive information directly from a parallel application execution, the behavior of a given resource, or the measurements of a computational probe. As of today, tests are being carried on with a basic implementation that collects information about resource utilization. These tests try to evaluate the implementation in order to define the latency of requests and response propagation through a given hierarchy structure. First results show that the approach is scalable, but further large-scale tests in the Grid'5000 must be performed to stress the implementation in situations where network latency and bandwidth are heterogeneous.

As stated in the introduction, this paper proposes the connection between the hierarchical collection implementation described here with the visualization tool Pajé. The component that makes this connection is described in next section.

4. DIMVisual Pajé Reader

DIMVisual [6] can be used to integrate traces collected by different tools and registered with specific file formats. Its implementation has several input modules that are specific to the traces that the modules can read. In its core, the DIMVisual performs a series of transformations in the trace data, especially the integration of the type hierarchy found in the input trace files and clock synchronization, for instance. The traditional output implementation of DIMVisual is to write a trace file in the Pajé format. Since this format is generic, almost all specific trace files can be easily mapped to the output format of DIMVisual.

The problem with the traditional approach of DIMVisual is that it obligates its users to pass through a file before analyze the information. The limitations imposed by this approach appear more clearly when trace files are large or the information needs to be analyzed in an online fashion. Since the hierarchical collection model presented in previous section already enables the online collection of data and centralization in the DIMVisual core, our main objective is the conception of an output module to DIMVisual that can send the integrated data directly to a visualization tool.

Since the initial implementation of DIMVisual already worked by generation trace files in the Pajé file format, suited for its visualization tool, we decided to maintain this link between DIMVisual and Pajé. For this reason, the module described here must be configured directly into the Pajé tool, as a different way of input data.

Figure 2 depicts the behavior of the DIMVisualReader and related components. The DIMVisual Integrator, which is part of the DIMVisual core's components, generates as output a flow of timestamped objects that represents the observed entities behavior. These objects are a high-level representation of Pajé events. The flow is received by the DIMVisualReader module, whose implementation follows the internal protocol of Pajé [2]. The responsibility of the

DIMVisualReader is to transform the flow of objects into textual representations using the Pajé file format. These representations are sent to the existing PajeEventDecoder filter, inside the Pajé tool, and then sent through the graph of Pajé components as in the usual configuration of the visualization tool.

The main benefit of coupling this module with the hierarchical collection model is that it enables the online visualization of the monitoring data collected by the system. This benefit influences directly the observability of the data, since it connects the visualizations provided by Pajé, and all its filters, to the hierarchical distributed collection model.

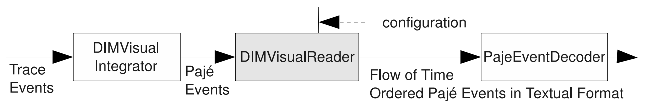

Figure 2. DIMVisualReader and related components.

The DIMVisualReader works based on the input data. Every time new trace information is available on its input, in the form of DIMVisual Pajé's objects, it converts to textual representations in its output. This behavior works well with the hierarchical collection model presented in previous section, since it acts based on the availability of new monitoring information.

The component is also configurable. It receives from the user interface or from another component a set of rules and configurations that affect its main behavior. A typical configuration for the module includes the specification of trace location data and synchronization information. With regard to the hierarchical collection model, the configuration may have further data that enable the component to connect to multiple data aggregators and collectors.

The DIMVisual Reader module is already in use in the Triva prototype [5], being used to connect trace events from KAAPI applications [3] directly to the visualization. As of today, the implementation of online visualization analysis is still not possible because the hierarchical model of previous section is not linked with DIMVisual core's components. However, as soon this task is accomplished, the online visualization is straightforward.

5. Conclusion

Distributed systems like Grids usually have monitoring information distributed across several resources. The data can be composed of traces from a parallel application, and also of dynamic information that represents the state of resources. In order to analyze this data, it is necessary to collect them in a scalable and dynamic way. The final objective of this collection is the centralization of the information so they can be analyzed together. The analysis task can be performed through a series of visualizations that show the behavior evolution of parallel applications and resources.

This paper presented a novel hierarchical distributed collection model to gather monitoring data from distributed resources. The model is composed of collectors, aggregators and clients, which form a hierarchical structure adaptable to resources needs and limitations. The model is implemented using the concepts of the DIMVisual integration tool. To enable the online visualization of the monitoring data collected by the hierarchical collection model, we also presented in the paper the DIMVisual Pajé Reader module. The combination of the hierarchical model and the DIMVisual reader module enables the online visualization of traces using Pajé, a generic visualization tool.

Future work includes the analysis of latency between the first collection of a monitoring data to its visualization. The prototype of the hierarchical model is currently under evaluation in clusters and in the Grid'5000. As soon this evaluation finishes, we intend to connect the output of the collection mechanism to the visualization.

References

[1] K. Czajkowski, C. Kesselman, S. Fitzgerald, and I. Foster. Grid information services for distributed resource sharing. In *HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, page 181, Washington, DC, USA, 2001. IEEE Computer Society.

[2] J. de Kergommeaux, B. Stein, and P. Bernard. Paje, an interactive visualization tool for tuning multi-threaded parallel applications. *Parallel Computing*, 26(10):1253–1274, 2000.

[3] T. Gautier, X. Besseron, and L. Pigeon. Kaapi: A thread scheduling runtime system for data flow computations on cluster of multi-processors. In *Proceedings of the international workshop on Parallel symbolic computation*, pages 15–23, New York, NY, USA, 2007. ACM.

[4] M. L. Massie, B. N. Chun, and D. E. Culler. The ganglia distributed monitoring system: Design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004.

[5] L. M. Schnorr, G. Huard, and P. O. A. Navaux. 3d approach to the visualization of parallel applications and grid monitoring information. In *Proceedings of the 9th IEEE/ACM International Conference on Grid Computing*, Washington, DC, USA, 2008. IEEE Computer Society.

[6] L. M. Schnorr, P. O. A. Navaux, and B. de Oliveira Stein. Dimvisual: Data integration model for visualization of parallel programs behavior. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, pages 473–480, Washington, DC, USA, 2006. IEEE Computer Society.