

CLUE : An Anomaly Detection System for PlanetLab

Mike Wawrzoniak
mhw@cs.princeton.edu

Abstract

As distributed systems continue to reach new levels of scale and complexity, they introduce new challenges to their control mechanisms. Detecting anomalous behavior of a distributed system can be the crucial element of the system's control strategy. This paper motivates and presents a design of an anomaly detection system currently being developed for the PlanetLab [10] distributed system and its services.

1 Motivation

Many distributed systems, such as PlanetLab [10], and its hosted services [12, 5, 11], individually, and collectively, continue to reach new levels of complexity. Every day, new code is added to the pool of running software that utilizes the evolving hardware environments. Paradoxically, these new, synthetic, man made systems, have reached the point of scale and of complexity, in their design and behavior, that they are now as unknown to humans as their organic counterparts. This real explosion of complexity poses an increasing challenge to the task of controlling these synthetic distributed organisms. One approach to address the problem is to endow these systems with an external control cycle, where the state of a system is monitored, its behavior is learned, and the most desirable control actions are inferred and executed.¹

From one perspective, every system, or a program in general, is such a control cycle, changing its state according to its previous state and the environment. However, programs and systems do go wrong, either acting contrary to their intended behavior or not in their interest,

¹Another alternative, ideologically opposite, is to (re)introduce determinism into the distributed systems, which, at a certain level, alleviates the problem of not knowing the current state of the system, and not knowing how the system may react to certain actions. In such a scenario, what is left is a search over the space of deterministic resource allocation strategies. However, it is historically clear, that the organic process of technology development and especially adaptation, cannot be constrained to such a 'dictatorial' form.

they encountered unanticipated internal states and unexpected changes in their environment. An external control cycle, that treats a system as a foreign mechanism implementing certain intentions, can add an additional layer of control mechanism, a *control overlay* that acts over the rationality explicitly expressed in the body of the system.

Implementing such a control cycle poses a number of challenges. Starting with system instrumentation, collection and possibly aggregation of sensory data, performing the appropriate analysis, finally to coordinating the execution of the selected actions. Constructing such adaptive systems in a challenge, this work specifically focuses on the initial process of system behavior analysis for the purposes of detecting anomalous system states. The process of collecting data and executing appropriate actions is outside of the context of this work, but it is the scope that informs a number of decisions made.

This study is in the context of the PlanetLab [10] system and the services [12, 5, 11] hosted by it. However, it is believed to have wider applicability. PlanetLab is a global platform for deploying and evaluating large scale distributed systems. It is a platform, currently composed of over 700 Internet connected nodes, spanning 336 sites in 35 countries. All of the resources are shared among PlanetLab users who are researchers developing and deploying large scale distributed systems. Many of the services are long-running and used by hundreds of thousands of users every day.

2 Requirements

This section, based on experiences with PlanetLab, outlines some of the requirements that an anomaly detection system must consider.

The initial goal of the PlanetLab anomaly detection system is to focus the management efforts on the elements of a distributed system that are potentially anomalous and may need correcting. In the context of PlanetLab, the system anomaly detection can be subdivided into two categories: the services hosted by PlanetLab and the Planet-

Lab infrastructure itself.

Developing and running PlanetLab services can be a challenging task. The services must handle real users and real world network traffic which may exercise unanticipated scenarios. Additional difficulty comes from the fact that the services run in the shared environment, where resource availability may depend on other services. This is a different environment from the usual dedicated platform model. Furthermore, the services often implement very complex logic which can significantly raise the level of difficulty for their robust implementation. Lastly, these complex and large scale systems are usually looked after by a very small group of researchers, who often may not know in what state is the service they are responsible for. Automated detection of anomalies of the services hosted on PlanetLab can be a valuable tool to the researchers managing and studying these systems, but the service anomalies may also have a negative impact on their environment - the network, and the PlanetLab platform, therefore this is an important tool for the PlanetLab platform administrators as well.

The PlanetLab platform itself is also a distributed system, and its elements also exhibit anomalous behavior, hardware faults, misconfiguration problems, changes in network characteristics, upgrade failures, etc. therefore monitoring of the platform system itself can also provide a significant benefit to PlanetLab management.

The behavior of PlanetLab distributed systems (both services, and the infrastructure) can be subdivided further into two categories : local behavior and global behavior. The local behavior corresponds to the behavior of the system at a particular point of presence, a concrete PlanetLab node. This element of the distributed system is not treated as a single system, working independently of the other instances, only reacting to its current state and environment. It is often the case that a distributed system is composed of a large number of similar (of only of few kinds) local instances, that behave in a similar way on all of the nodes. Because of this, behavior of one instance can inform the normal behavior of other instance of the system, and therefore, the learned behavior patterns of one element can be compared with behaviors of others. This characteristic allows for learning a more complete picture of functioning of local elements, as the same local system encounters different environment at different locations. Information learned about a unique behavior

of local instance on one node can be used to analyze a behavior of a local instance at a different node even if it never was encountered at that node, allowing for faster and more complete behavior model learning.

A global behavior of a distributed system is the behavior defined by the collective combination of local behaviors of its components. Ultimately, this is the most important type of a behavior of a distributed system. As the global environment changes so does the global behavior, systems reconfigure, shift resources, manipulate redundancy to provide the intended services. It is crucial to detect anomalies at the global level, as well as at the local level.

To move beyond just the detection of the unusual behavior, the anomaly detection system may leverage existing service management systems or expert knowledge to determine if certain behavior is normal or not. However, such information may be partial, only some of the behaviors may be covered by the existing infrastructure, or expensive, in the case of contacting a human expert. Because of this, the system must deal with incompletely labeled behavior history and must do its best to infer semantic similarities between system states. If the learning requires all of the observed states to be labeled for training, the system loses its value as an aid to the existing management infrastructure, or becomes a burden of human experts, whom in the end it is to assist.

The amount of sensory data that is produced by a real running large scale distributed system is huge. It must be filtered and sampled at the appropriate points and granularity, however, even then the semantically valuable data set is too large to collect it, store it, and perform post-processing. The anomaly detection system must therefore be able to deal with a streaming sensory data, and process it as it arrives, minimizing the size of archive necessary in order to function.

Valuable information of the system behavior arises from correlation of multiple sources of information concurrently. Measuring one metric may reveal major problems, but subtle behaviors are only signaled through combination of multiple metrics. Therefore the system must deal not with each source of information separately, but it must be able to correlate a large number of dimensions concurrently, to learn the system behavior in more detail.

The behavior of the studied systems is not uniform over time. In addition to reacting to fluctuating work-

loads, the systems are also regularly maintained, upgraded, restarted, exchanging control information etc. The anomaly detection system must be able to learn the temporal behavior of the system.

In summary, the anomaly detection system must address heterogeneous local and global system behaviour, must work with partially labeled system states, must be able to deal with continuously streaming large volume of multidimensional data, and learn system temporal behavior.

3 Learning System Behavior

This section describes how the Clue anomaly detection system approach satisfies the requirements proposed in the previous section.

In the Clue system, the process of anomaly detection is treated as a probabilistic classification problem [4] of the observed system behavior. Based on system observations, models of the behavior are constructed. Using the learned behavior models, all new system behavior observations are classified as either *normal*, *abnormal*, or *new*. The *normal* label is assigned to behaviors that matches (are sufficiently similar to) previously recognized correct states of the system, the *abnormal* label corresponds to behaviors that matches previously recognized incorrect system states, and the *new* label corresponds to system behavior that has not been seen before, and therefore can be either normal or abnormal [7].

In the event of a new system behavior detected, domain knowledge of the system (human or programmatic expert) is consulted to provide the appropriate classification. The expert feedback is then incorporated into the models, and it is used for further classification. When a component of the system enters a state that is classified as *abnormal* the system's administrators are automatically contacted so that appropriate actions can be applied and feedback collected.

Using the appropriate sensory data of the monitored distributed systems is crucial to learning the behavior. The Clue system uses time series sensory data from PlanetLab. The multidimensional stream is composed of vectors describing local service behavior on all nodes for all active services on PlanetLab. In addition, the information on the behavior of the local nodes is collected as well. It is used

to build both platform models, as well as for parameters for the service models. The stream of information on the global state of the system is synthesized from the stream of local states, and then used to train the global models and classify global system behavior.

In addition to the collected sensory data, the behavior models rely on domain knowledge to label behaviors as normal and abnormal. These labels are acquired either from a human expert, through a web interface, or pragmatically, using a domain specific, expert specified control service.

The multidimensional sensory data collected is of different formats. Some of the vector elements are continuous variables, some are multinomial, some are just binary. To deal with the heterogeneous data types, probabilistic graphical models [6] are used. This allows for composition of different types of variables as one joint probability distribution. In addition, the use of graphical models allows to leverage some of the independence properties between variables to reduce the number of parameters in the behavior models. Variables that have temporal properties are projected onto directional coordinates [8].

A system element may have multiple behaviors that are normal, and multiple behaviors that are abnormal. For this reason, the behavior models are hierarchical graphical models [6] to allow learning multiple distinct behaviours per class. (For example, the `princeton_codeen` service uses most of the nodes as forwarding proxies, however a small subset of the nodes are only performing infrastructure maintenance tasks, which exhibit different behavior.)

Only small subset of the sensory data is labeled as normal or abnormal, most of the data is unlabeled. The system must leverage the unlabeled data to extract more complete behavior models. This is addressed by using a semi-supervised learning [2, 3] of the hierarchical graphical mixture model, which allows learning the models leveraging both labeled and unlabeled data.

Since the amount of the sensory data that is to be collected and processed is very large, the models are fitted using on-line learning methods. This allows the system to consume all of the sensory data immediately, and discard most of it immediately. The only data points that are stored are points representing new and anomalous behavior (this is a result of classification), and data points that could provide valuable labels to the model (e.g. data point for a normal but rare behavior).

Some of the observed behaviors are relatively rare, which causes avoiding overfitting difficult. Employing Bayesian approach [1] improves the ability to model such behaviors by using priors over the model parameters (the prior distributions can also be initialized using knowledge acquired from similar systems to the one being observed).

In summary, the behavior model employed by the Clue anomaly detection system is a on-line, semi-supervised, hierarchical, bayesian graphical model for classification.

4 Initial Experience

An early version of the discussed system is already functioning and operating non-stop for three months, as of writing this text. It continuously collects and analysis data for a subset of slices and contacts their administrators when anomalies are detected. There is a wide range of possibilities to instrument PlanetLab and its services, each with its possible advantages and costs. Depending on the kind and specificity of the data, different types of behaviors can be distinguished and different resolutions of anomalies can be detected. As the starting point, the current implementation of the Clue system utilizes the PlanetLab resource utilization data available through CoMon [9]. The data specifies the resource utilization of each service for all nodes, and the total resource utilization for all of the PlanetLab nodes. The data is sampled at 5 minute intervals and is collected in a centralized archive. Different service behaviors exhibit different resource utilization, generate different traffic patterns, use different amounts of memory, therefore this is a reasonable starting point, but will be expanded further in the future.

The currently used data parameters of each service for each node are as follows :

Transmit bandwidth: Bandwidth transmitted by the service in the past 1 and 15 minutes, in Kb/s.

Receive bandwidth: Bandwidth received by the service in the past 1 and 15 minutes, in Kb/s.

Process count: Number of processes currently used by the service.

Physical memory: Amount of physical memory (in MB) used by the service.

Virtual memory: Amount of virtual memory (in MB) allocated by this service.

CPU share: Fraction of the CPU of the node currently consumed by the service.

Memory share: Fraction of memory of the node currently consumed by the service.

Ports used: Number of network ports used by the service.

The domain knowledge feedback is provided through programmatic and web² interfaces, that allow domain experts to label system behaviors, which are then used to learn the classification models. The implementation of the models described is only partial, however it already proved to be a useful tool to find a number of true problems in distributed systems monitored. As the implementation becomes more complete, and the system behaviors are modeled in more detail, the ability to detect anomalies is expected to become more accurate.

References

- [1] C. M. Bishop. *Pattern Recognition and Machine Learning*. 2006.
- [2] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.
- [3] F. Cozman, I. Cohen, and M. Cirelo. Semi-supervised learning of mixture models and bayesian networks, 2003.
- [4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.
- [5] M. Freedman, E. Freudenthal, and D. Mazieres. Democratizing content publication with coral, 2004.
- [6] M. I. Jordan. Graphical models. *Statistical Science*, 19, 140-155, 2004.
- [7] M. Lauer. A mixture approach to novelty detection using training data with outliers. *Lecture Notes in Computer Science*, 2167:300–314, 2001.
- [8] K. V. Mardia and P. E. Jupp. *Directional Statistics*. Wiley Series in Probability and Statistics, 2000.

²For an example of princeton_codeen service http://clue.cs.princeton.edu/clue/?slicename=princeton_codeen

- [9] K. Park and V. S. Pai. CoMon: A mostly-scalable monitoring system for PlanetLab. *Operating Systems Review*, 40(1), January 2006.
- [10] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir. Experiences building planetlab. In *Proceedings of the 7th USENIX Symposium on Operating System Design and Implementation (OSDI '06)*, Seattle, WA, November 2006.
- [11] S. Rhea, B. Godfrey, B. Karp, J. Kubiawicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. Opendht: A public dht service and its uses, 2005.
- [12] L. Wang, K. Park, R. Pang, V. S. Pai, and L. Peterson. Reliability and security in the CoDeeN content distribution network.