Monitoring distributed computing beyond the traditional time-series histogram

M S Doidge , *P A* Love[*], and *J* Thornton

Department of Physics, Lancaster University, LA1 4YB, UK

Abstract. In this work we describe a novel approach to monitor the operation of distributed computing services. Current monitoring tools are dominated by the use of time-series histograms showing the evolution of various metrics. These can quickly overwhelm or confuse the viewer due to the large number of similar looking graphs. We propose a supplementary approach through the sonification of real-time data streamed directly from a variety of distributed computing services. The real-time nature of this method allows operations staff to quickly detect problems and identify that a problem is still ongoing, avoiding the case of investigating an issue a-priori when it may already have been resolved. In this paper we present details of the system architecture and provide a recipe for deployment suitable for both site and experiment teams.

1 Introduction

This work continues the development of a novel monitoring system for real-time distributed computing operations. As distributed computing systems continue to grow in complexity and scope we argue the benefits of a monitoring system beyond the ubiquitous time-series histogram. High energy physics experiments have large teams of people operating the computing infrastructure and there is significant effort spent of monitoring the systems and addressing detected service problems. A popular development trend is to consolidate this effort by centralising the monitoring infrastructure and leverage the experience of others, as done by the Worldwide LHC Computing Grid (WLCG) collaboration [1]. We mention some negative aspects of this approach as motivation to explore other novel monitoring systems. In this case a sonification architecture will be described along with some perceived benefits to operating distributed computing systems.

[*] Corresponding author : p.love@lancaster.ac.uk

2 Beyond time-series histograms

The scale and complexity of monitoring systems increases in step with the systems being monitored. As these activities grow so does the operation and maintenance effort. Cooperation between experiments within WLCG is encouraged and the sharing of monitoring infrastructure has been a subject for this cooperation. An example of this shared infrastructure is the MONIT deployment at CERN [2]. This system relies heavily on time-series histograms to display the changing value of a large number of metrics.

Time-series histograms are ubiquitous in most monitoring systems as they are easy to interpret by human operators. However with the large scale deployment of centralised monitoring services we are seeing a number of issues with the monitoring system itself such as:

1. missing data - where a time period bin shows null values for particular metric due to an outage of the collection agent.

2. stale data - where the time period shown is not current and usually due to a broken messaging system.

3. incorrect data - when discrepancies are found the trust in the infrastructure is difficult to regain.

4. user fatigue - a human operator examine dozens of these plots which require high levels of concentration.

How can we address these problem? Our approach is to focus on real-time monitoring and situational awareness. With this in mind we developed a sonification tool where message streams are rendered in real-time. The lack of historic views is acknowledged but the benefit is a high level of trust in the output and allows for immediate feedback on faults, as well as immediate feedback when solutions are deployed. Another aspect of time-series histograms is user fatigue when processing these graphs wheras the use of sonification to render the output is known to be less intrusive [3] allowing the operators focus on different tasks whilst still responding to system faults.

For further information about the field of sonification we recommend *The Sonification Handbook* [4] as a reference.

3 Architecture and implementation

The sonification tool described here is called Subtlenoise [5][6] and the latest architecture is shown in Figure 1. where the collection and transport of event messages are shown. The origin of events is diverse so we use a flexible approach to send raw messages to a central process. The central process is implemented using the Mosquito Broker [7] where UDP packets from collection agents (A,B,C) are published to message channels on the broker. The core Subtlenoise orchestration script 'dj.py' is a Python program which subscribes to the Mosquito feed and consumes the raw message content. The program translates the message into an Open Sound Control (OSC) message which is sent to the audio rendering engine. It is this stage where the interpretation occurs and transforms the raw message into a suitable sound event. The current audio rendering system is the SonicPi [8] platform running on a Raspberry Pi hardware with loud speaker.

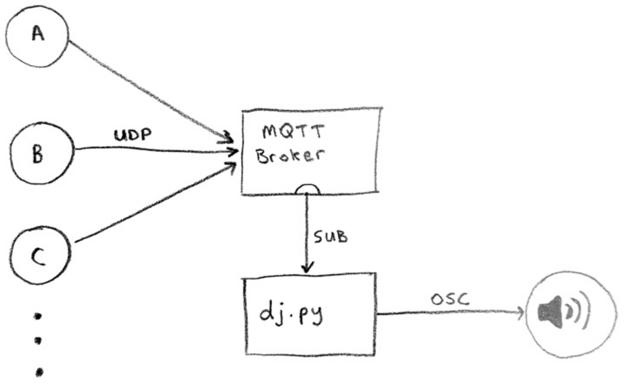

Fig. 1. Message flow architecture in Subtlenoise where A, B, C are collection agents which send raw messages to a central broker. The python script dj.py transform these messages into OSC commands suitable for audio rendering..

This work is very much an exploration of the concept and as such we chose the simplest tools available. The use of UDP over TCP was intentional as it provides a lightweight non-intrusive mode of operation where TCP connection handling does not. It is extremely important that monitoring tools do not cause issues with the application they are purporting to monitor. Due to the nature of sonification the possible loss of UDP packets would not affect the outcome, and if the packet loss was significant it would actually provide useful information about network degradation in the form of silence. This is one of the main features of sonification, the lack of noise is itself informative.

4 Results

4.1 Analysis and interpretation of data streams

We currently use a parameter mapping scheme [4] to transform the raw message into OSC commands suitable for audio rendering. This is the simplest approach to

sonify a data stream in contrast to a model-based scheme where an external model is used to dynamically determine the audible output. The main output attributes of a note are pitch, duration, cut-off, and pan. We can use any of these attributes as a way to distinguish the incoming raw data. For example, raw messages with numerical content can be mapped to pitch in a number of predefined bins. Taking the case of a compute job which has a clear run-time duration we can use this duration mapped onto note duration so short jobs are short notes and long jobs are sustained notes. The effect becomes obvious when faulty compute jobs are present because the short duration notes are easy to hear.
Another mapping example is when the raw message feed contains a string field with a small number of distinct values. In the case of WLCG computing this could be the "site name" and we can map this onto the Pan attribute. The effect of this is each WLCG site gets mapped to a particular location in the stereo field, some sites on the left channel, others on the right channel, and others somewhere in between.
To help define the rendering scheme we developed an analysis tool where a particular message stream was analyzed and each field was characterized in terms of its domain and frequency. Each message was tokenized and fields were ranked in terms of usefulness to the audio attributes. Invariant fields were ignored and domain range of numerical fields were used to set the scale of output values. This analysis was done for each source of data.

4.2 A range of use-cases

A number of use-cases have been tested with the Subtlenoise system and the automated analysis of the message content has been adequate to produce a distinct audible rendering. These examples include messages from compute jobs, data transfers, and storage system logs. Each has a distinct rendering which provide useful insight to the health of the different systems.
The most explored use case is the monitoring of ATLAS [9] pilot jobs where every job sends two messages, one at the start and one on completion whether this a clean exit or a known fault case. The message rate was about 50Hz and the components operated as expected without issue. We found the audible output was a good balance of subtle noise which wasn't distracting, but at the same time it highlighted operational issues which needed attention.

4.3 Benefit of audio-rendered monitoring

Working with this system to sonify real-world applications has highlighted a few clear benefits over visual representation of the same metrics.

1 Service feedback is immediate and real-time which allow human operators to quickly conclude a service is healthy.

2 Auditory displays have high levels of trust. If the monitoring chain has a problem then no sound is heard which avoids the visual problem of looking at stale information.

3 Subtle nuances in the service performance can be identified by small changes in the audio stream. These can help pre-empt service faults rather than reacting to existing problems.

5 Conclusion

The Subtlenoise project provides a novel way to monitor distributed computing operations in real-time. The use of sonification allows an intuitive way to interpret message streams produced by the various services which constitute large and complex computing infrastructure.

References

1. P Saiz et al. "WLCG monitoring Consolidation and further evolution" Journal of Physics: Conference Series **664** (2015) 062054 doi:10.1088/1742-6596/664/6/062054
2. A Aimar et al. "MONIT: Monitoring the CERN Data Centres and the WLCG Infrastructure" EPJ Web of Conferences **214**, 08031 (2019) doi: 10.1051/epjconf/201921408031
3. Saskia Bakker et al., "Knowing by ear: leveraging human attention abilities in interaction design" *J Multomodal User Interfaces* (2012) 5:197-209 DOI: 10.1007/s12193-011-0062-8
4. The Sonification Handbook http://sonification.de/handbook
5. P A Love 2015 *J. Phys.: Conf. Ser.* **664** 062034 doi:10.1088/1742-6596/664/6/062054
6. Subtlenoise project page https://github.com/ptrlv/subtlenoise
7. R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," *The Journal of Open Source Software*, vol. 2, no. 13, May 2017, doi: 10.21105/joss.00265
8. Sonic Pi - The Live Coding Music Synth for Everyone. https://sonic-pi.net/
9. The ATLAS Collaboration 2008 The ATLAS Experiment at the CERN Large Hadron Collider *JINST* **3** S08003