Scaling the PuNDIT project for wide area deployments

Scaling the PuNDIT project for wide area deployments

Shawn McKee[1], Jorge Batista[1], Gabriele Carcassi[1], Constantine Dovrolis[2], Danny Lee[2]

[1] Randall Laboratory, Physics Department, University of Michigan, 450 Church Street, Ann Arbor, Michigan, 48109-1040, USA

[2] Klaus Advanced Computing Building, College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, Georgia, 30332-0765, USA

E-mail: `smckee@umich.edu`

Abstract. In today's world of distributed scientific collaborations, there are many challenges to providing reliable inter-domain network infrastructure. Network operators use a combination of active monitoring and trouble tickets to detect problems, but these are often ineffective at identifying issues that impact wide-area network users. Additionally, these approaches do not scale to wide area inter-domain networks due to unavailability of data from all the domains along typical network paths. The Pythia Network Diagnostic InfrasTructure (PuNDIT) project aims to create a scalable infrastructure for automating the detection and localization of problems across these networks.

The project goal is to gather and analyze metrics from existing perfSONAR monitoring infrastructures to identify the signatures of possible problems, locate affected network links, and report them to the user in an intuitive fashion. Simply put, PuNDIT seeks to convert complex network metrics into easily understood diagnoses in an automated manner.

We present our progress in creating the PuNDIT system and our status in developing, testing and deploying PuNDIT. We report on the project progress to-date, describe the current implementation architecture and demonstrate some of the various user interfaces it will support. We close by discussing the remaining challenges and next steps and where we see the project going in the future.

1. Introduction

The PuNDIT project started in 2014 as an NSF funded satellite project of the Open Science Grid[1]. Its goal is to provide a framework allowing near real-time analysis of perfSONAR[2] network measurements to both identify network problems and localize them within the various measured network paths.

The paper will cover the current status and challenges being faced by the project.

2. The PuNDIT Project

The PuNDIT project has already been described in a previous CHEP paper[3] and we refer the reader to it for background. Figure 1 presents a high-level view of PuNDIT. PuNDIT is creating an infrastructure that can analyze perfSONAR network metrics (primarily latency and trace-route) to both identify specific network problems and correlate those problems to localize their origin in the network. This capability is one of the most requested features from

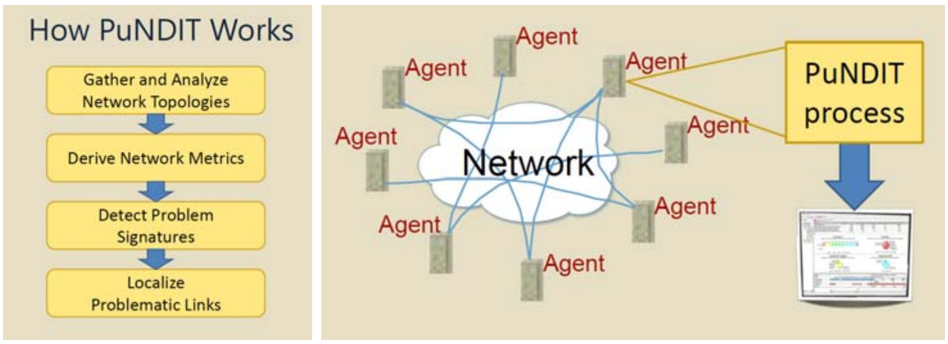


Figure 1. Shows a high-level view of how PuNDIT works. The left side shows the basic workflow and the right side illustrates the system architecture for a typical deployment.

site administrators who have deployed perfSONAR for WLCG[4] and OSG. Potential users of PuNDIT want a system watching their networks and alerting them to problems when they develop.

3. Architecture

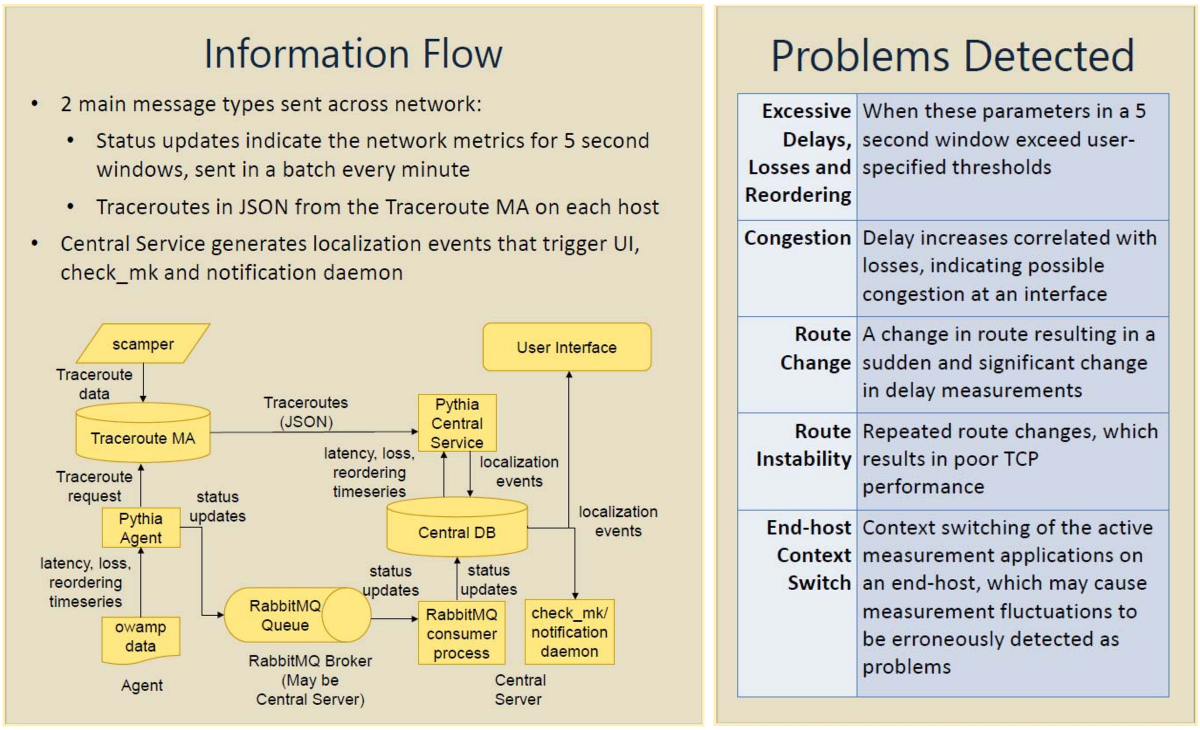
To meet its goals the PuNDIT system requires a number of components:

- An agent for each perfSONAR toolkit instance that will participate in a given PuNDIT deployment.
  - The agent must be able to access perfSONAR data metrics without disrupting the primary goal of making reliable network measurements.
  - The agent must process the data to identify signatures of network problems.
  - The agent must send the results of its analysis to a central PuNDIT service which provides the tomographic analysis[5] necessary for a given PuNDIT deployment to locate the origin of the network problems identified.
- A central PuNDIT server responsible for gathering and analyzing the results of a set of PuNDIT agents.
  - The server must provide a reliable, secure way for agents to communicate their results to the server.
  - The server must take the network problems and network path information received and provide the tomographic analysis required to localize the origin of the problems observed.
  - The server needs to provide a user interface to provide access the the measurements and analysis results.

The project team has worked on developing, tuning and packaging two sets of software: the agent and the server. In Figure 2 we show both how information flows through the system and what types of problem signatures PuNDIT is capable of identifying.

A critical part of the architecture is the agent-server communication component. Initial prototyping simply used remote access directly to a central MySQL database running on the PuNDIT server node. In addition to the serious security concerns this raises, the performance and reliability were not sufficient for a real-world deployment. Since the perfSONAR developer team was already exploring the use of RabbitMQ[6], we selected it as our means of providing agent to server communication. RabbitMQ is an implementation of messaging based upon AMQP[7]

From the RabbitMQ features page: "*Messaging enables software applications to connect and scale. Applications can connect to each other, as components of a larger application, or to user*



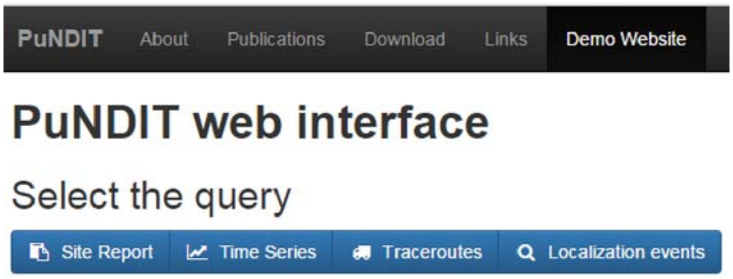
Information Flow

- 2 main message types sent across network:
  - Status updates indicate the network metrics for 5 second windows, sent in a batch every minute
  - Traceroutes in JSON from the Traceroute MA on each host
- Central Service generates localization events that trigger UI, check_mk and notification daemon

Problems Detected

Excessive Delays, Losses and Reordering	When these parameters in a 5 second window exceed user-specified thresholds
Congestion	Delay increases correlated with losses, indicating possible congestion at an interface
Route Change	A change in route resulting in a sudden and significant change in delay measurements
Route Instability	Repeated route changes, which results in poor TCP performance
End-host Context Switch	Context switching of the active measurement applications on an end-host, which may cause measurement fluctuations to be erroneously detected as problems

Figure 2. The left side illustrates the architecture of the information flow through PuNDIT while the right side lists the type of network problems PuNDIT is sensitive to.



PuNDIT web interface

Select the query

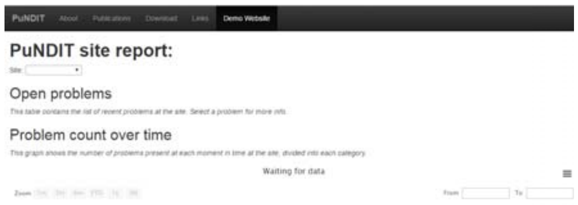
Figure 3. The top-level PuNDIT GUI web page showing the current four sub-options available: Site Report, Time Series, Traceroutes and Localization Events.

devices and data. Messaging is asynchronous, decoupling applications by separating sending and receiving data." This is exactly what we needed to do for PuNDIT and RabbitMQ has given us a simple-to-use way to efficiently and reliably communicate between our PuNDIT agents and server.

4. Graphical User Interface Design and Implementation

The PuNDIT central server needs an effective user interface to allow PuNDIT users the ability to see network problems and drill-into the details. We have constructed a graphical user interface (GUI) to be the primary means users will interact with the system. Underlying the GUI is a database.

The database consists of two parts. In the first set of tables the PuNDIT server places the data as it is collected. This facilitates the implementation of the server, which only needs to



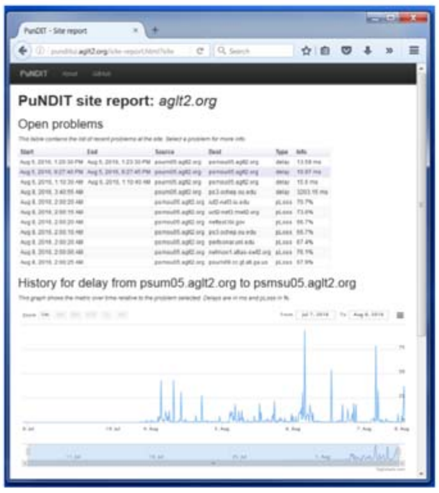
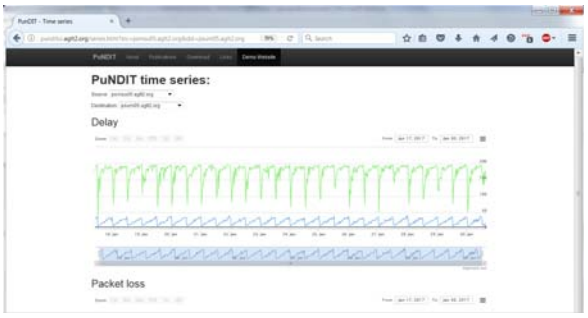

PuNDIT - Site report
PuNDIT site report: aglt2.org
Open problems
History for delay from psum05.aglt2.org to psmsu05.aglt2.org

Figure 4. This shows the Site Report page with a drop-down to select the site (on the left) and how the user can drill into a specific problem after selecting a site (on the right).



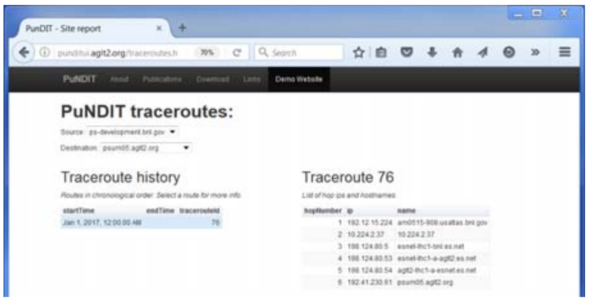


Figure 5. An example of the PuNDIT GUI showing a timeseries measurement between two hosts on the left and and traceroute example on the right.

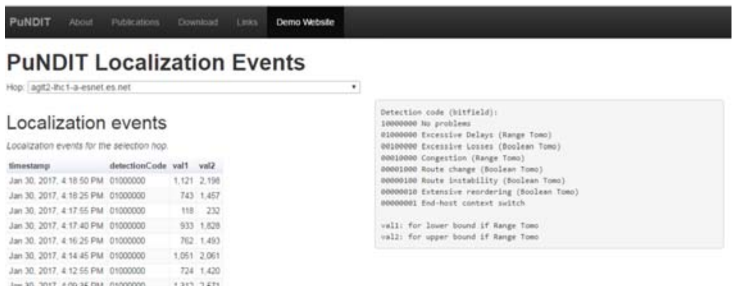
write new entries without having to manage the relationships between database entities.

The second set of tables are de-normalized and designed to provide better read performance. A script running every 30 seconds takes care of moving the data from the staging tables to the permanent tables. The same script also aggregates problems, trace-routes and time series into time periods so that the data is in a format more readily consumable by the client.

The web based user interface works directly off this second set of tables. A Java server application monitors the database and transmits the data through WebSockets[8] as it updates. The configuration of the server application defines which queries are made available to the client and at what update rate. The web pages use HTML classes and attributes to define UI component that connect to the server through WebSockets. The components are expanded on the client by a JavaScript library that takes care of connecting them to the right data channels, updating the UI with the incoming data and broadcasting to the server the user selection.

The technology used to move the data from the database to the UI is designed to easily accommodate changes in database schema and/or changes in the UI layout and logic. This allows to iterate on both to tweak and adapt the implementation as user feedback is gathered.

The resulting current user interface top-level page is shown in Figure 3 and provides 4 options to choose from as described in the caption. Figures 4, 5 and 6 show some details for each of these options.



PuNDIT Localization Events

Hop: agit2-lhc1-a-esnet.es.net

Localization events

Localization events for the selection hop.

timestamp	detectionCode	val1	val2
Jan 30, 2017, 4:18:50 PM	01000000	1,121	2,198
Jan 30, 2017, 4:18:25 PM	01000000	743	1,457
Jan 30, 2017, 4:17:55 PM	01000000	118	232
Jan 30, 2017, 4:17:40 PM	01000000	933	1,828
Jan 30, 2017, 4:16:25 PM	01000000	762	1,493
Jan 30, 2017, 4:14:45 PM	01000000	1,051	2,061
Jan 30, 2017, 4:12:55 PM	01000000	724	1,420

```
Detection code (bitfield):
10000000 No problems
01000000 Excessive Delays (Range Tomo)
00100000 Excessive Losses (Boolean Tomo)
00010000 Congestion (Range Tomo)
00001000 Route change (Boolean Tomo)
00000100 Route instability (Boolean Tomo)
00000010 Extensive reordering (Boolean Tomo)
00000001 End-host context switch

val1: for lower bound if Range Tomo
val2: for upper bound if Range Tomo
```

Figure 6. This shows a simulated set of localization results from PuNDIT.

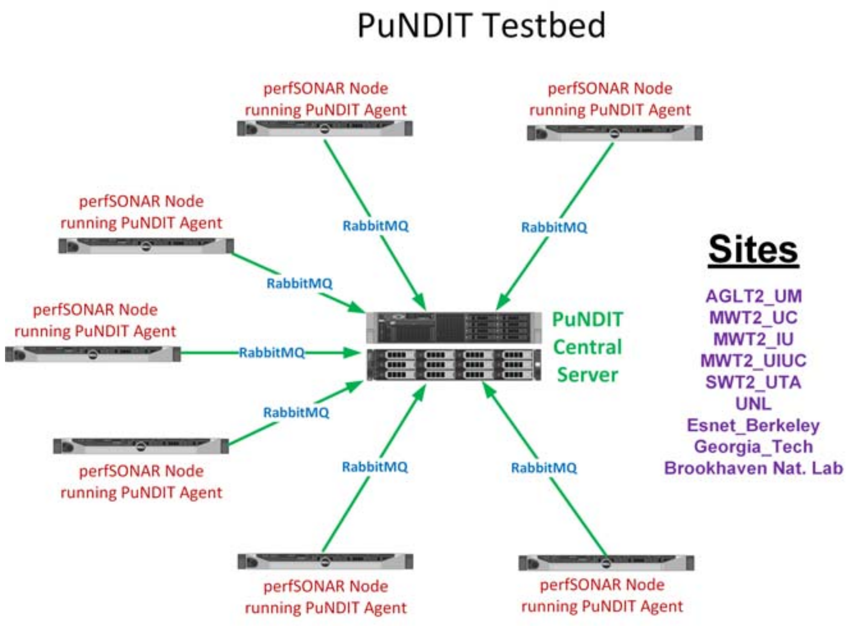

PuNDIT Testbed
perfSONAR Node running PuNDIT Agent
perfSONAR Node running PuNDIT Agent
perfSONAR Node running PuNDIT Agent
perfSONAR Node running PuNDIT Agent
perfSONAR Node running PuNDIT Agent
perfSONAR Node running PuNDIT Agent
perfSONAR Node running PuNDIT Agent
RabbitMQ
RabbitMQ
RabbitMQ
RabbitMQ
RabbitMQ
RabbitMQ
RabbitMQ
PuNDIT Central Server
Sites
AGLT2_UM
MWT2_UC
MWT2_IU
MWT2_UIUC
SWT2_UTA
UNL
Esnet_Berkeley
Georgia_Tech
Brookhaven Nat. Lab

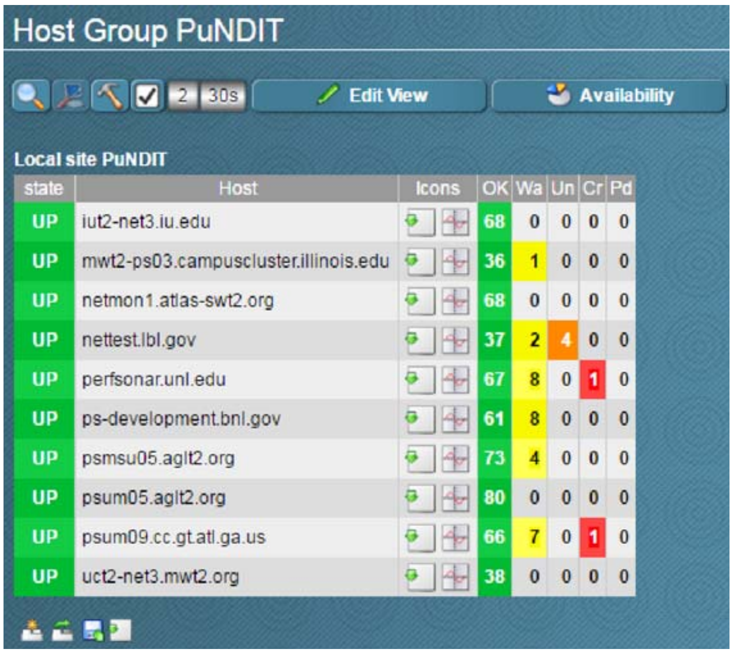
Figure 7. A diagram of the PuNDIT testbed we are using to develop and prototype the PuNDIT system. The participating sites are shown on the right.

5. Deployment and Testing

We have set up a national scale testbed to work on developing and prototyping the PuNDIT system using the OSG perfSONAR testbed. This testbed was created as part of the original USATLAS work in testing perfSONAR's suitability for monitoring our networks. As perfSONAR nodes were replaced or updated, the older nodes were kept in use to run specific tests and to vet release candidates from the perfSONAR team. The PuNDIT project asked sites with testbed nodes if they would be willing to serve as part of a PuNDIT testbed and, as shown in Figure 7, nine agreed.

The testbed has been critical for both testing PuNDIT at reasonable scale and to perform additional testing under diverse, real-world conditions. Each site has different hardware and is connected to different networks in different parts of the country spanning from east coast to west coast and north to south.

During testing we have run into a number of issues that the testbed helped identify. Originally we had a site with a perfSONAR host with only 2GB of RAM. This turned out to be insufficient to run both PuNDIT and perfSONAR reliably at scale and we will have to require a minimum of 4GB of RAM to deploy PuNDIT.



Local site PuNDIT

state	Host	Icons	OK	Wa	Un	Cr	Pd
UP	iut2-net3.iu.edu		68	0	0	0	0
UP	mwt2-ps03.campuscluster.illinois.edu		36	1	0	0	0
UP	netmon1.atlas-swt2.org		68	0	0	0	0
UP	nettest.lbl.gov		37	2	4	0	0
UP	perfsonar.unl.edu		67	8	0	1	0
UP	ps-development.bnl.gov		61	8	0	0	0
UP	psmsu05.aglt2.org		73	4	0	0	0
UP	psum05.aglt2.org		80	0	0	0	0
UP	psum09.cc.gt.atl.ga.us		66	7	0	1	0
UP	uct2-net3.mwt2.org		38	0	0	0	0

Figure 8. The Check_mk PuNDIT testbed monitoring group.

To track the impact of PuNDIT on it host systems and to monitor and manage the state of the testbed, we have chosen to use Check_mk[9]. Having a central Check_mk server for PuNDIT has allowed us to easily track our testbed hosts, perfSONAR and PuNDIT services.

In Figure 8 you can see the monitoring group for the PuNDIT testbed. The Check_mk server maintains a set of graphs for every monitored host and service; hovering over the graph items shows a preview plot while clicking takes you to a detailed set of graphs with varying time-scales. Clicking on a particular hostname allows you to drill into details for that host's services as shown in Figure 9.

6. Next Steps

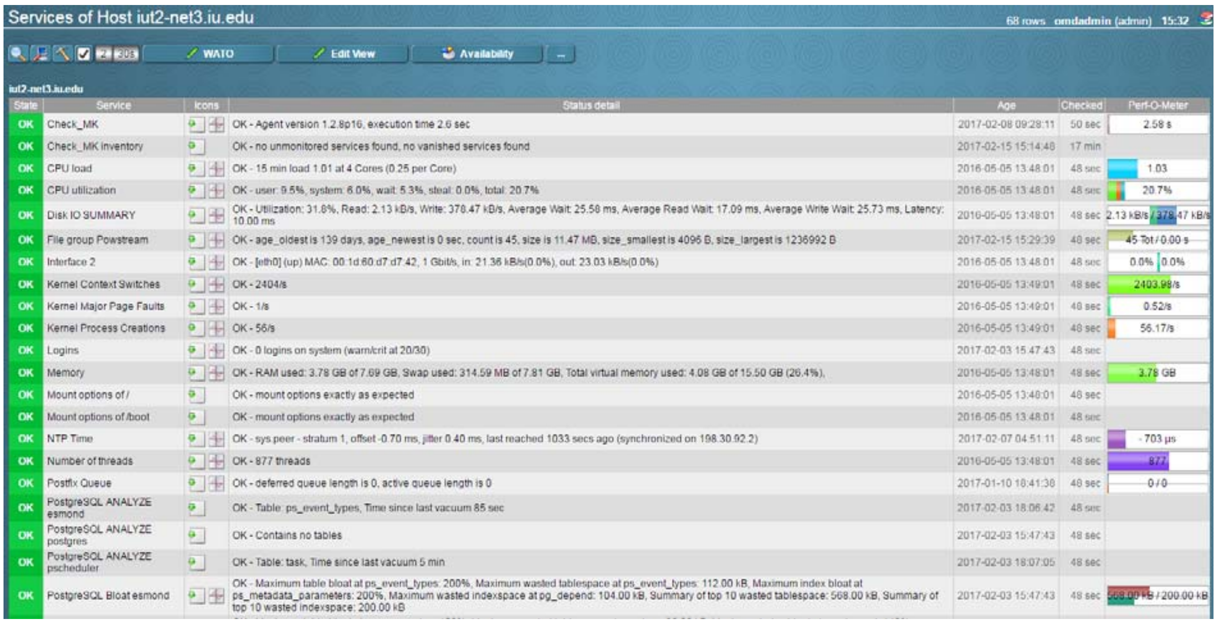
To provide a working set of software in the form of two RPMS we still have a number of things to do. In this section we will describe the work remaining to deliver software for the broader community to use. At a high-level there are three main areas of work:

- Testing and optimization of the agent-server interaction
- Optimizing the graphical user interface
- Packaging and documenting the agent and server software

We currently have working agent and server code but it needs to be suitably adapted to the upcoming perfSONAR Toolkit v4.0 release. There have been extensive changes to perfSONAR since the PuNDIT project began and the changes coming in v4.0, especially the switch to pScheduler, require some significant changes in how the PuNDIT agent will access the data. In general these changes make it much more straightforward to program within the perfSONAR context but it still takes a bit of work to refactor, implement and test the needed changes.

7. Conclusions

The PuNDIT project still has a significant set of hurdles to cross before it can deliver a usable system for near real-time network analysis and problem location. In the final few months of the


Services of Host iut2-net3.iu.edu
68 rows omdadmin (admin) 15:32
2
30s
WATO
Edit View
Availability
...
iut2-net3.iu.edu

State	Service	Icons	Status detail	Age	Checked	Perf-O-Meter
OK	Check_MK		OK - Agent version 1.2.8p16, execution time 2.6 sec	2017-02-08 09:28:11	50 sec	2.58 s
OK	Check_MK inventory		OK - no unmonitored services found, no vanished services found	2017-02-15 15:14:48	17 min	
OK	CPU load		OK - 15 min load 1.01 at 4 Cores (0.25 per Core)	2016-05-05 13:48:01	48 sec	1.03
OK	CPU utilization		OK - user: 9.5%, system: 6.0%, wait: 5.3%, steal: 0.0%, total: 20.7%	2016-05-05 13:48:01	48 sec	20.7%
OK	Disk IO SUMMARY		OK - Utilization: 31.8%, Read: 2.13 kB/s, Write: 378.47 kB/s, Average Wait: 25.58 ms, Average Read Wait: 17.09 ms, Average Write Wait: 25.73 ms, Latency: 10.00 ms	2016-05-05 13:48:01	48 sec	2.13 kB/s / 378.47 kB/s
OK	File group Powstream		OK - age_oldest is 139 days, age_newest is 0 sec, count is 45, size is 11.47 MB, size_smallest is 4096 B, size_largest is 1236992 B	2017-02-15 15:29:39	48 sec	45 Tot / 0.00 s
OK	Interface 2		OK - [eth0] (up) MAC: 00:1d:60:d7:d7:42, 1 Gbit/s, in: 21.36 kB/s(0.0%), out: 23.03 kB/s(0.0%)	2016-05-05 13:48:01	48 sec	0.0% 0.0%
OK	Kernel Context Switches		OK - 2404/s	2016-05-05 13:49:01	48 sec	2403.98/s
OK	Kernel Major Page Faults		OK - 1/s	2016-05-05 13:49:01	48 sec	0.52/s
OK	Kernel Process Creations		OK - 56/s	2016-05-05 13:49:01	48 sec	56.17/s
OK	Logins		OK - 0 logins on system (warn/crit at 20/30)	2017-02-03 15:47:43	48 sec	
OK	Memory		OK - RAM used: 3.78 GB of 7.69 GB, Swap used: 314.59 MB of 7.81 GB, Total virtual memory used: 4.08 GB of 15.50 GB (26.4%),	2016-05-05 13:48:01	48 sec	3.78 GB
OK	Mount options of /		OK - mount options exactly as expected	2016-05-05 13:48:01	48 sec	
OK	Mount options of /boot		OK - mount options exactly as expected	2016-05-05 13:48:01	48 sec	
OK	NTP Time		OK - sys.peer - stratum 1, offset -0.70 ms, jitter 0.40 ms, last reached 1033 secs ago (synchronized on 198.30.92.2)	2017-02-07 04:51:11	48 sec	-703 µs
OK	Number of threads		OK - 877 threads	2016-05-05 13:48:01	48 sec	877
OK	Postfix Queue		OK - deferred queue length is 0, active queue length is 0	2017-01-10 18:41:38	48 sec	0 / 0
OK	PostgreSQL ANALYZE esmond		OK - Table: ps_event_types, Time since last vacuum 85 sec	2017-02-03 18:06:42	48 sec	
OK	PostgreSQL ANALYZE postgres		OK - Contains no tables	2017-02-03 15:47:43	48 sec	
OK	PostgreSQL ANALYZE pscheduler		OK - Table: task, Time since last vacuum 5 min	2017-02-03 18:07:05	48 sec	
OK	PostgreSQL Bloat esmond		OK - Maximum table bloat at ps_event_types: 200%, Maximum wasted tablespace at ps_event_types: 112.00 kB, Maximum index bloat at ps_metadata_parameters: 200%, Maximum wasted indexspace at pg_depend: 104.00 kB, Summary of top 10 wasted tablespace: 568.00 kB, Summary of top 10 wasted indexspace: 200.00 kB	2017-02-03 15:47:43	48 sec	568.00 kB / 200.00 kB

Figure 9. An example of drilling into a specific testbed host on Check_mk. For legibility the lists of services is truncated.

project we hope to be able to wrap up the remaining tasks discussed in the previous section and deliver a basic but functional pair of RPMS that can be used to construct a PuNDIT system.

Should PuNDIT prove useful to one or more communities there are a number of improvements and extensions that we already know would be beneficial, e.g., IPv6 support, other operating systems besides RHEL/CentOS 6/7, better topology representation, GUI improvements and adding an analysis capability for bandwidth tests. In such a case we hope to either submit a follow-on proposal or assemble the user community to provide the effort required to make those improvements and extensions. If the original team is unable to continue working on PunDIT, all code is being hosted in GitHub[10] and could become the source for either the perfSONAR team to take over or for the Open Source community to start from.

Acknowledgments
The authors would like to acknowledge the contributions of the perfSONAR project, namely staff from ESnet, GEANT, Indiana University, Internet2, the University of Michigan and all the contributors shown at http://www.perfsonar.net/about/who-is-involved/.

We would also like to acknowledge and thank all the perfSONAR testbed sites who volunteered to host our prototype PuNDIT services on their systems.

Lastly we gratefully acknowledge the National Science Foundation (grants 1440571 and 1440585) which supported the work in this paper.

References
[1] Altunay M, Avery P, Blackburn K, Bockelman B, Ernst M, Fraser D, Quick R, Gardner R, Goasguen S, Levshina T, Livny M, McGee J, Olson D, Pordes R, Potekhin M, Rana A, Roy A, Seghal C, Sfiligoi I and Würthwein F 2011 A science driven production cyberinfrastructure: the open science grid vol 9 (Springer Netherlands) pp 201–218 ISSN 1570-7873 URL `http://dx.doi.org/10.1007/s10723-010-9176-6`

[2] Hanemann A, Boote J, Boyd E, Durand J, Kudarimoti L, apacz R, Swany D M, Trocha S and Zurawski J 2005 Perfsonar: A service oriented architecture for multi-domain network monitoring vol 3826 (Springer Berlin Heidelberg) pp 241–254 URL `http://link.springer.com/chapter/10.1007%2F11596141_19`

[3] Batista J, Dovrolis C, Lee D and McKee S 2015 *Journal of Physics: Conference Series* **664** 052027 URL http://stacks.iop.org/1742-6596/664/i=5/a=052027
[4] Bird I 2011 Computing for the large hadron collider vol 61 (Annual Reviews) pp 99–118 URL http://www.annualreviews.org/doi/abs/10.1146/annurev-nucl-102010-130059
[5] Ma L, He T, KLeung K, Towsley D and Swami A 2013 Efficient identification of additive link metrics via network tomography (IEEE) URL http://ieeexplore.ieee.org/abstract/document/6681627/
[6] Rabbitmq open source project URL https://www.rabbitmq.com/
[7] Apache activemq URL http://activemq.apache.org/
[8] Websockets overview URL https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
[9] Check_mk an extension to the nagios monitoring system URL https://en.wikipedia.org/wiki/Check_MK
[10] Pundit project source code URL https://github.com/pundit-project