Architecture and Implementation of a 25000 FPS Radio Camera on the Long Wavelength Array

Karthik Reddy[a], Judd D. Bowman[a], Jayce Dowell[b], Greg B. Taylor[b], Adam P. Beardsley[c], and Craig Taylor[b]

[a]School of Earth and Space Exploration, Arizona State University, Tempe, AZ 85287, USA
[b]Department of Physics and Astronomy, University of New Mexico, Albuquerque, NM 87131, USA
[b]Physics Department, Winona State University, Winona, MN 55987, USA

ABSTRACT

A highly optimized E-field Parallel Imaging Correlator (EPIC), currently under commissioning on the Long Wavelength Array in Sevilleta, New Mexico, can image the sky at a rate of 25,000 FPS per polarization and frequency. The system consists of six processing nodes, each producing images of the visible sky with a 1-degree spatial resolution at an 80 ms temporal resolution, covering a 3.2 MHz spectral window below 100 MHz, yielding a total bandwidth of 19.2 MHz. Light curves for selected sources of interest will be extracted from each image into a distributed database, and 5-minute accumulations are archived on the disk for further analysis. In this paper, we describe the components of our real-time imaging system, designed as a plug-and-play solution to deploy EPIC on similar arrays with only minor modifications.

Keywords: Real-time radio imaging, Long Wavelength Array, EPIC, System Architecture

1. INTRODUCTION

Studies of extreme transient phenomena like the Fast Radio Bursts (FRBs) in the dynamic radio sky are at the forefront of modern radio astronomy. Their need for wide imaging fields and high sensitivity have driven radio telescopes towards dense layouts with several antennas, and advances in general purpose Graphics Computing Units (GPUs) allowed for achieving high bandwidth and time resolution [1, for example]. However, traditional correlators, which compute the integrated pair-wise antenna cross-correlations to form images of the sky, become computationally expensive to implement on modern telescopes with hundreds to thousands of antennas as their complexity scales as the square of the number of antennas. Hence, an efficient processing algorithm that produces an identical scientific output is desirable.

EPIC is a direct imaging algorithm designed to produce full-sky observations at a high cadence using telescopes with dense but otherwise arbitrary layouts.[2] Unlike traditional correlators, EPIC directly transforms antenna electric fields into sky images. That means the computational complexity decreases from an extreme N_A^2 to a gentle $N_g \log_2 N_g$ where N_A is the number of antennas and N_g is the size of the aperture plane grid or equivalently the number of pixels in the output image.[3]

We selected the Long Wavelength Array in Sevilleta,[4] New Mexico (LWA-SV), observing in the 3-88 MHz range to deploy EPIC to detect FRBs below 100 MHz through continuous sky-monitoring. LWA-SV is an excellent system for our purpose as it contains 256 dipole antennas arranged in a dense configuration within a 100m region. The F-Engine at this station applies a Fourier transform to digitized electric field time series from each antenna in 40μs windows, yielding a channel width of 25 KHz. That means the correlator must generate 25000 images of the visible sky per second for each frequency and polarization to operate in real-time. The F-Engine also divides the observing bandwidth of about 20 MHz into 6 subbands, each with 132 channels (3.3 MHz), and multicasts the raw channelized data over a 100G ethernet link that will be utilized by the EPIC imaging system.

In the development phase, EPIC was initially implemented in Python[2] and was later ported to GPUs to be run on a single node at LWA-SV.[5] The GPU code generated full-sky full-polarization images in real-time

Software and Cyberinfrastructure for Astronomy VIII, edited by Jorge Ibsen, Gianluca Chiozzi, Proc. of SPIE Vol. 13101, 131011C · © 2024 SPIE
0277-786X · doi: 10.1117/12.3012524

with a ≈4 deg spatial resolution (32 sq. pixels) covering a bandwidth of 2.25 MHz and integrated to yield a time resolution of about 80 ms. Although further rounds of optimizations, including utilizing the Bifrost stream processing framework, allowed increasing the image size to 64 sq. pixels with a spatial resolution of ≈2 deg,[6] the bandwidth was limited to 1.8 MHz per GPU and allowed imaging a single polarization. We identified several bottlenecks through extensive profiling that prompted a rebuild of the imaging code[7] from scratch with state-of-the-art features in NVIDIA's GPU programming platform. The new imager[8] generates 128 sq. pixel full-sky full-polarization images of the sky with a spatial resolution of ≈1 deg, covering a 3.2 MHz bandwidth on a single GPU, which although is slightly smaller than the desired 3.3 MHz. That means we effectively generate 12.8 million full-sky images per second across all frequencies and polarizations per GPU. They are integrated in time to yield typical 40-80 ms time-resolutions.

We are deploying the optimized EPIC imager on six GPU nodes at LWA-SV to continuously monitor the sky for transient events with a bandwidth of 19.2 MHz. With an 80 ms cadence, the imager will produce about 110 TB of images per day that will be reduced to analyze the spectral nature of sources of interest. Operating a monitoring system with this throughput requires significant computing resources with reliable and high-performance software pipelines.

Here, we describe the architecture of our EPIC-based real-time full-sky monitoring system, code-named "Alchemist", that is being deployed at LWA-SV. Section 2 provides an overview of the system requirements of the Alchemist, section 3 provides the architecture and implementation details, and 4 summarizes the deployment and describes a way forward.

2. SYSTEM REQUIREMENTS

Full-sky imaging capability allows simultaneous monitoring of an arbitrary number of sources in the sky. Hence, although the imaging system is primarily aimed at detecting FRBs below 100 MHz, we will also monitor the sky for other sources of emission, including pulsars, the Sun and Jupiter, and meteors. The Alchemist will follow up FRB event triggers from multiple high-frequency radio sky monitors, including the Canadian Hydrogen Intensity Mapping Experiment (CHIME), the Deep Synoptic Array (DSA), and the Realfast project running commensally on the Jansky Very Large Array (JVLA), and gravitational wave events from the Laser Interferometer Gravitational-Wave Observatory (LIGO).

We will initially monitor a selected list of pulsars, FRBs, and the sun and later expand the monitoring to the entire visible sky with a real-time transient detection pipeline. Storing the images at the desired cadence of 80 ms is challenging, requiring about 110 TB per day. To enable rapid follow-up and analysis of events from sources of interest and to alleviate the storage requirements, light curves for each source can be extracted from the pixels of output images and ingested into a database. A 25 FPS video of the sky can be streamed to an online platform to support quick look-up. Longer accumulations can be archived on the disk to retain imaging data for future analyses. With an 80 ms cadence and assuming at least six sources are visible in the sky at any time, the six-node EPIC imager will yield at least ∼1 billion rows (∼1 TB) of light curve data daily. In addition, a storage space of about 300 GB per day is required to store 3-minute image accumulations and video streams. The 80 ms cadence light curves will be stored for 6 months from extraction, while longer accumulations and video streams will be permanently stored and made available to the community on a web platform.

Operating and maintaining a high throughput imager described above needs robust pipelines capable of handling large data streams with minimal latency. Using extensive software instrumentation to monitor the system will allow us to identify bottlenecks, optimize the system performance, and alert operators to resolve issues, reducing downtime quickly. Horizontal scalability is another key aspect that allows the expansion of computational and storage resources to accommodate higher spatial and temporal resolutions. In addition to these performance requirements, we are developing the Alchemist with a modular architecture that allows the deployment of an EPIC imager on any telescope array, requiring only minor changes to the configuration. The deployment scripts are built using Ansible and will be open-sourced.

3. SYSTEM ARCHITECTURE AND IMPLEMENTATION

We adopted a client-server model to implement Alchemist where a subset of nodes in the cluster act as coordinators or servers that designate work to the remaining worker or client nodes. Scalability is built into this model as all nodes work independently, and increased performance can be achieved by adding more nodes to the cluster. Furthermore, this model allows centralized management and control of the system. Figure 1 shows the architecture of Alchemist that is under deployment at LWA-SV. Although our setup includes one server node with six client nodes, we only show the setup for a single client server as the configuration is identical for the remaining nodes. Below, we describe the system's individual components and provide the deployment status.

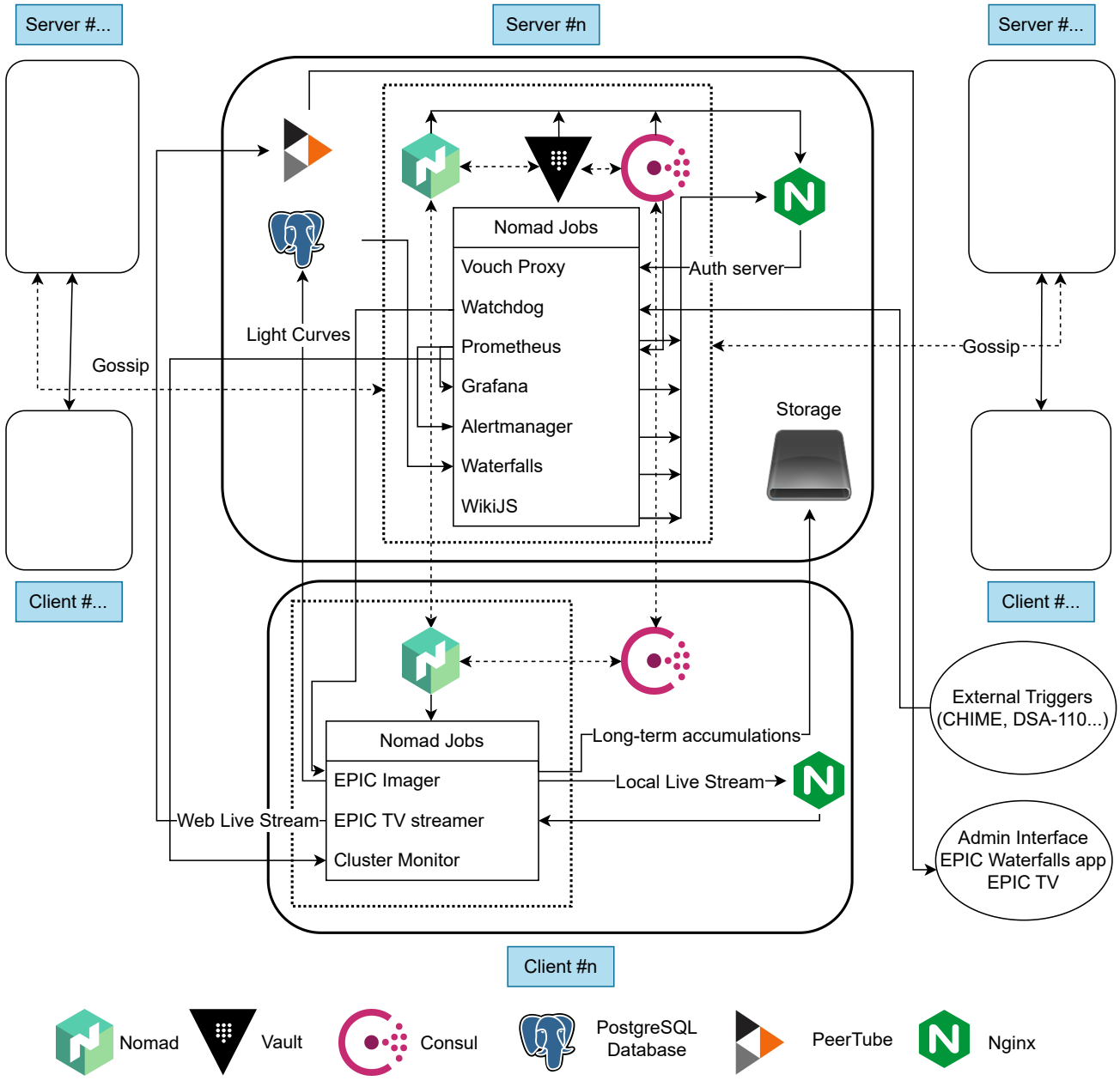

Figure 1. System architecture with individual components for the EPIC-based continuous sky monitor on LWA-SV.

3.1 Data Storage

We use the PostgreSQL database to store light curves of transient sources. Pixel values from a 5x5 grid centered around each source on output images are inserted into the database as individual rows in binary format (postgres BYTEA type). With an 80 ms cadence, we insert about 7 million rows per hour for each source. To minimize insertion and query times, we split the table into multiple partitions based on the timestamp, with each partition covering a period of 1 hour. We use a Postgres extension called `pg_partman`[9] to automate creating partitions periodically. We store 3-minute accumulations in fits format and single-channel quick-look images in PNG format on a 100 TB network-attached storage. We also store the associated metadata for each image in the database to facilitate querying and downloading data. In addition to high cadence light curves and long-term accumulations, the imager also generates a 25 FPS video of the sky using the FFmpeg[10] library. The videos are streamed to an instance of PeerTube,[11] an open-source video streaming platform, and are publicly available for live streaming. Replays are posted every 24 hrs and are saved to the network drive.

3.2 Nomad, Consul, and Vault

Alchemist comprises several processes, like imaging, monitoring, and live streaming, distributed across all nodes in the cluster. We use HashiCorp's open-source tools, namely Nomad,[12] Consul,[13] and Vault,[14] to deploy and manage these processes. Nomad is a flexible and scalable cluster manager and scheduler. It orchestrates deploying and managing containerized and standalone applications responsible for various stages in our sky monitoring pipeline. To enable robust service discovery and coordination within the cluster, we integrate HashiCorp Consul. Consul provides a dynamic, service-oriented architecture, where each service involved in the sky-monitoring pipeline can be easily discovered and accessed by other services. It also offers health checking and service monitoring features, ensuring that each service is operating correctly and allowing for automatic failover and recovery, thereby enhancing the reliability of the overall system. Finally, we integrate HashiCorp Vault to manage secrets and protect sensitive information. Vault provides a secure method for storing and accessing secrets like credentials and API keys required by the applications running within the cluster.

Each tool in the HashiCorp stack, namely Nomad, Consul, and Vault, is offered as a standalone binary run with configuration files written using HashiCorp Language (HCL). Nomad and Consul processes on each node, commonly called agents, can be run in client and server modes. In the server mode, Nomad schedules tasks on the client nodes, monitors their resource usage, and maintains the overall state of the cluster. Nomad servers also expose a user interface (UI) to manage the cluster interactively. Nomad clients receive task assignments from the servers and execute them. They report tasks' health and resource usage back to the servers, allowing servers to make informed scheduling decisions. Consul agents operate similarly. Consul servers store key-value data and service catalogs and handle service registration and de-registration, enabling clients to discover all services running in the cluster. Consul clients forward service discovery requests to the servers, perform health checks on local services, and report the results to the servers. Unlike Nomad and Consul, Vault lacks distinct client and server modes. Instead, Vault agents are run on multiple nodes that each maintain a copy of the secrets and authorized services in the cluster, which are the Vault clients, can retrieve them.

In our deployment at LWA-SV, we run Nomad and Consul in client mode on all client nodes and in server mode on the server node. Nomad client mode is also enabled on the server node to support running tasks described below. After one or more consul server agents are initialized, all Nomad, Consul, and Vault agents can auto-discover themselves using the service discovery feature provided by Consul. That means we can add new compute nodes to the cluster on-site and off-site without modifying any configuration files. Vault always starts in an uninitialized or sealed state where the master key that decrypts the secrets is encrypted. This encrypted master key is split into five parts; at least three keys are required to unseal the Vault. Hence, to prevent Vault from being stuck in a sealed state due to unscheduled node restarts, we store these encrypted keys in a secondary Vault cluster running on a mutually exclusive set of nodes that auto-unseals the primary Vault. All nomad agents are provided an access key to the Vault, allowing tasks to access the required secrets.

3.3 Monitoring and Alerting

We use Prometheus[15] to monitor the overall performance of the system. It collects (scrapes) metrics on regular intervals from HTTP endpoints exposed by processes (exporters) and ingests them into a time series database

(TSDB). The endpoints must be passed in a configuration file to Prometheus for initialization. However, many processes are required to run sky monitoring (see section 3.4), and manually adding them to the configuration files can introduce errors. Hence, we configure Prometheus servers to auto-discover new and existing processes that expose metrics endpoints through Consul and scrape metrics from them. Prometheus also analyzes real-time and historical data to generate alerts based on pre-defined rules. These alerts are sent to an instance of AlertManage[16]r that broadcasts them to a slack channel for quick resolution of issues. Finally, we use Grafana[17] to visualize metrics scraped by Prometheus. Grafana provides several interactive visualizations, such as time series plots and gauges, that can be organized into dashboards. We created multiple dashboards to track the computing, storage, and network resource usage of all nodes in the cluster. Figure 2 shows a GPU monitoring dashboard with visualizations tracking various properties of a GPU like fan speed, memory usage, and power usage that runs the imaging pipeline.

Figure 2. Grafana dashboard showing metrics collected by Prometheus from GPU exporter on a client node executing the real-time imaging pipeline.

3.4 Nomad Tasks

We divide the workload of Alchemist into multiple processes distributed across the cluster. Nomad launches these processes or jobs using configuration files written in HCL. Each job contains one or more tasks that are executed using task drivers. For our deployment, we use docker and `raw_exec` drivers that allow launching docker containers and standalone executables, respectively. Tasks are launched as long-running services or are scheduled periodically. We also register all services on Consul to allow them to discover each other, eliminating the need to set up accesses between them manually. Below, we describe each job orchestrated by the Nomad cluster at LWA-SV and indicate the types of nodes (client and server) where the job is launched in parentheses.

Watchdog (Server): This job runs a gRPC-based[18] watchdog server that maintains a list of sources of interest. It exposes endpoints to add new sources or query existing sources in the watch list. In the current deployment phase, the watch list only consists of a small set of bright pulsars, the sun, and sources from the A-team. Future versions of the watchdog will add the ability to dynamically add sources based on external triggers like CHIME.
EPIC Imager (Client): The job launches an imaging pipeline on each node that transforms raw data received from the telescope array into images. It queries the watchdog on regular intervals to fetch a watchlist and filters sources that fall within the current field of view. Light curves for these sources are extracted from the output images and are ingested into a Postgres database. Database credentials are fetched from Vault, and the pipeline configuration is fetched from the Consul key-value store. Nomad always restarts the pipeline with a change in any of these parameters. This allows us to control individual pipelines with minimal downtime. The imager streams a 25 FPS video to an RTMP endpoint hosted on a local NGINX server. It also exposes a metrics endpoint that tracks the execution times of each stage in the imaging pipeline and registers itself with Consul for use by Prometheus.

EPIC Streamer (Client): The streamer job exports live streams from local NGINX instances to the PeerTube instance running on the server node. The streamer is configured to restart every 24 hours to trigger PeerTube to post a replay of the stream over the past 24 hours. Figure 3 shows a replay of a video stream of the sky at 34.8 MHz on the EPIC TV website. The streamer job fetches PeerTube live stream keys from Vault.

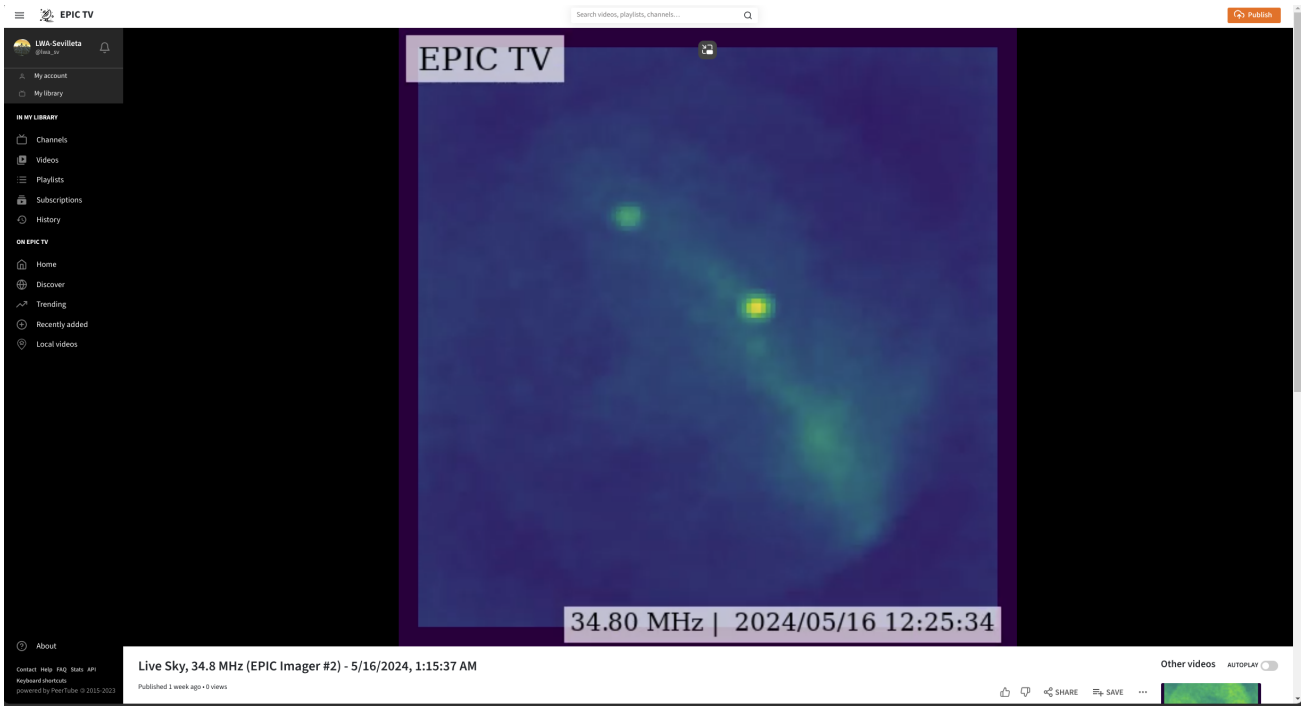

Figure 3. EPIC TV website (available at https://livetv.epic-astronomy.org replaying a video of the sky from an imager operating at 34.8 MHz. The Milky Way is seen as a bright stripe across the video, and the bright objects from top to bottom are Cas A and Cygnus A, respectively. This video also depicts the capability of EPIC to monitor any number of sources in the sky simultaneously.

Cluster Monitor (Client and Server): This job is executed on all nodes and launches four Prometheus exporters: NVIDIA GPU exporter, systemd exporter, node exporter, and Postgres exporter. Metrics collected by these exporters are visualized on Grafana dashboards (see fig. 2, for example).
Vouch Proxy (Server): This job launches Vouch Proxy, which runs a single sign-on solution for NGINX. We configure it to use a Github team as an authentication method for the administration UIs exposed by Nomad, Consul, Vault, Prometheus, and AlertManager. **Prometheus, AlertManager and Grafana (Server):** We launch each tool described in section 3.3 as a separate job on the server. Prometheus auto-discovers all the metrics endpoints available in the cluster using consul. The Alertmanager, which broadcasts alerts to a slack channel, fetches the authorization key from Vault.
WikiJS (Server): This job hosts an internal wiki with detailed documentation on Alchemist. WikiJS is an open-source wiki platform that supports hosting and adding documentation in multiple formats, including HTML, plain text, and markdown. It stores all the data in a Postgres database where the credentials are fetched from Vault.
Waterfalls (Server): This job hosts a web application called Waterfalls that allows interactively exploring live and historical spectrograms for all sources in our watchlist. Figure 4 shows the visualization interface, including a spectrogram of the Sun during a flaring event. The front end for this web application is built using Nuxt UI web components and is launched using a NodeJS-based server. The backend server is built using FastAPI and exposes HTTP endpoints based on the Open API specification to access data from the light curve database.

Figure 4. Waterfalls web application interface (available at https://epic-astronomy.org/waterfalls/spectrograms/lwasv), showing a spectrogram of the Sun during a flaring event on May 10.

4. SUMMARY

In this paper, we describe Alchemist, a full-sky monitoring system aimed at detecting FRBs, currently under deployment on LWA-SV. The underlying EPIC-based imager can generate 25000 full-sky images per second, frequency, and polarization on a single GPU and are integrated to produce images with an 80 ms time resolution. Light curves for selected sources are ingested into a partitioned Postgres database, allowing efficient storage and retrieval. Imagers also export 25 FPS video streams of the sky to a publicly available PeerTube instance. We are monitoring a set of pulsars, FRBs, and the Sun in the current deployment. Future deployments will allow monitoring the entire visible sky and responding to triggers from other facilities such as CHIME and DSA-110.

The cluster comprises one coordinator node, and six other compute nodes, each processing a bandwidth of 3.2 MHz to yield a total coverage of 19.2 MHz below 100 MHz. The system uses HashiCorp's Nomad, Consul, and Vault to deploy and manage processes in the cluster. We elected HashiCorp's stack due to its horizontal scalability and centralized management. Nomad cluster integrates with Consul and Vault to orchestrate the deployment and management of processes responsible for different segments in the sky-monitoring system. Furthermore, configuration for the imaging pipelines is stored in Consul's key-value store, allowing us to dynamically modify imaging parameters with minimal downtime. We use Prometheus, AlertManager, and Grafana to monitor the system's health. Prometheus auto-discovers metrics endpoints exposed by services using Consul, eliminating the need for manual configuration. It also sends alerts to AlertManager, which broadcasts them to a Slack channel for rapid resolution. We developed Alchemist as a modular and scalable system that can be deployed on any radio telescope array with dense and compact layouts, requiring only minimal changes to the configuration.

REFERENCES

[1] Denman, N., Amiri, M., Bandura, K., Connor, L., Dobbs, M., Fandino, M., Halpern, M., Hincks, A., Hinshaw, G., Höfer, C., et al., "A gpu-based correlator x-engine implemented on the chime pathfinder," in [*2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*], 35–40, IEEE (2015).

[2] Thyagarajan, N., Beardsley, A. P., Bowman, J. D., and Morales, M. F., "A generic and efficient e-field parallel imaging correlator for next-generation radio telescopes," *Monthly Notices of the Royal Astronomical Society* **467**(1), 715–730 (2017).

[3] Morales, M. F., "Enabling next-generation dark energy and epoch of reionization radio observatories with the moff correlator," *Publications of the Astronomical Society of the Pacific* **123**(909), 1265 (2011).

[4] Taylor, G., Ellingson, S., Kassim, N., Craig, J., Dowell, J., Wolfe, C., Hartman, J., Bernardi, G., Clarke, T., Cohen, A., et al., "First light for the first station of the long wavelength array," *Journal of Astronomical Instrumentation* **1**(01), 1250004 (2012).
[5] Kent, J., Dowell, J., Beardsley, A., Thyagarajan, N., Taylor, G., and Bowman, J., "A real-time, all-sky, high time resolution, direct imager for the long wavelength array," *Monthly Notices of the Royal Astronomical Society* **486**, 5052–5060 (05 2019).
[6] Krishnan, H., Beardsley, A. P., Bowman, J. D., Dowell, J., Kolopanis, M., Taylor, G., and Thyagarajan, N., "Optimization and commissioning of the EPIC commensal radio transient imager for the long wavelength array," *Monthly Notices of the Royal Astronomical Society* **520**, 1928–1937 (Apr. 2023).
[7] Reddy, K., Bowman, J. D., Beardsley, A. P., Taylor, G. B., Dowell, J., and Taylor, C., "An optimized gpu kernel for real-time radio imaging," in [*2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*], 523–527 (2023).
[8] EPIC Collaboration, "EPIC: Real-time imaging pipeline for the long wavelength array." https://github.com/epic-astronomy/LWA_EPIC (2024).
[9] Osumi, K., "pg_partman." https://github.com/pgpartman/pg_partman (2024). Software.
[10] Tomar, S., "Converting video formats with ffmpeg," *Linux Journal* **2006**(146), 10 (2006).
[11] Framasoft, "Peertube." https://joinpeertube.org/ (2024). Software.
[12] HashiCorp, "Nomad." https://www.nomadproject.io/ (2024). Software.
[13] HashiCorp, "Consul." https://www.consul.io/ (2024). Software.
[14] HashiCorp, "Vault." https://www.vaultproject.io/ (2024). Software.
[15] The Prometheus Authors, "Prometheus." https://prometheus.io/ (2024). Software.
[16] The Prometheus Authors, "Alertmanager." https://prometheus.io/docs/alerting/latest/alertmanager/ (2024). Software.
[17] Grafana Labs, "Grafana." https://grafana.com/ (2024). Software.
[18] The gRPC Authors, "grpc." https://grpc.io/ (2024). Software.