Scaling Orbit Propagation Analysis Capabilities with Cloud Computing, Data Analytics and the Unified Data Library

Derek Chen, Chang Zhang, Mark Mendiola, Ann Chervenak, Alex Gonring, Jeffery Won, Scott Bergonzi, Vincent Kong, Eltefaat Shokri
The Aerospace Corporation

ABSTRACT

As the number of space objects proliferates, there is a growing need to perform automated, high fidelity orbit propagation on a large number of these objects. We present an approach that performs high fidelity orbit propagation at large scale on a private or public cloud. We download state vector data generated by LeoLabs for objects in low earth orbit from the Unified Data Library (UDL); we perform orbit propagation on these state vectors by running multiple instances of the Aerospace Corporation's TRACE (Trajectory Analysis and Orbit Determination Program) software on a cloud using the Aerospace Data Exploitation (DEX) analytics platform. We describe the architecture and implementation of this system and explain how this approach can be used to track thousands of space objects automatically and efficiently.

1. INTRODUCTION

There is a growing need to perform orbit propagation on a large number of space objects automatically and with high fidelity. The number of commercial and government satellites and other space objects that need to be tracked is increasing rapidly. There are now frequent launches of dozens of satellites, whose trajectories and orbits need to be tracked from launch to continuous operation. Ventures such as SpaceX Starlink, Amazon Kuiper, and OneWeb have begun launching hundreds of low earth orbit (LEO) satellites that will be used to provide global internet and communication capabilities. Tracking all these space objects is a daunting challenge, since much of this tracking is currently initiated manually using mature, legacy orbit determination applications that run on dedicated computing resources with limited user access. To meet the need to scale the number of space objects for orbit propagation as well as the need to provide broader access to the user community, we propose to automate and parallelize these applications using modern cloud computing infrastructure.

We present an approach that performs high fidelity orbit propagation at large scale on a cloud. We download LeoLabs [1] state vector data from the Unified Data Library (UDL) [2] and perform orbit propagation on these state vectors by running multiple instances of the Aerospace Corporation's TRACE Trajectory Analysis software [3] on a private cloud using the Aerospace Data Exploitation (DEX) analytics platform [4]. In this presentation, we describe the architecture of the system and explain how this approach can be used to track thousands of space objects automatically and efficiently.

Our work utilizes state vectors generated by LeoLabs, a company that operates multiple radar stations around the globe and tracks thousands of objects in Low Earth Orbit (LEO). LeoLabs publishes observations from their radar stations, state and uncertainty estimations for each tracked object, and detailed sensor information to the Unified Data Library (UDL). The UDL is a repository for space situational awareness data from many sources and is being developed by BlueStaq for the Air Force Research Laboratory (AFRL) and Air Force Space and Missile Systems Center (SMC).

Our orbit propagation is done using TRACE, the Aerospace Corporation's Trajectory Analysis and Orbit Determination Program. TRACE has been developed and maintained since the 1960s and is used in the design and analysis of problems associated with high-accuracy orbital trajectory modeling, ephemeris and parameter estimation, and related error analysis. TRACE provides high-fidelity orbit propagation, orbit determination, and covariance analysis. The application is currently implemented in FORTRAN with some modules in C. TRACE typically runs on a single computer under the Windows or Linux operating systems.

We run these orbit propagation analyses on the DEX Data Analytics Platform, a reference architecture and prototype implementation for an evolvable, responsive, and cloud-based data exploitation platform [4]. DEX was developed as

part of the Enterprise Ground Service (EGS) to support satellite operations and Space Domain Awareness. The platform extracts and delivers actionable information by running configurable, general-purpose and domain-specific analytics on heterogeneous data sets. In this presentation, we will describe the architecture of the DEX Data Analytics Platform and explain how its capabilities were used by the TRACE team to easily package and deploy their software.

We will describe in detail the end-to-end orbit propagation workflow that runs on the DEX cloud-based data analytics platform and supports automated, large scale, parallel analysis of many objects in congested, contested space. This work is sponsored by SMC/ECXC, Space C2 and the Kobayashi Maru Program Office. This work supports program office efforts in the areas of Space Domain Awareness (SDA) and data management.

2. TRACE

TRACE is The Aerospace Corporation's trajectory analysis and orbit determination program [3]. Originally developed in the 1960s, TRACE has been used regularly to support a variety of high-fidelity analyses. It has been and continues to be used to model and evaluate proposed improvements to the Global Positioning System (GPS) Next Generation Operational Control System (OCX) [5]. Additionally, TRACE has been used to validate various contractor tools and orbit propagators.

Orbit propagation and ephemeris generation is one key function of TRACE which was selected for use with the DEX platform. Additionally, TRACE capabilities include measurement data generation, measurement error modeling, orbit determination, and covariance analysis [6]. A consistent feature of TRACE across all modes of use is the level of detail and configurability available for the user. Users have the ability to model orbital trajectories with a high-order numerical integrator or a selection of analytic propagators. Use of the numerical integrator allows the user to configure many detailed force models to appropriately characterize the dynamic environment for their scenario. Output files provide detailed ephemerides and intermediate results of the user's choosing, including states in multiple coordinate frames, partial derivatives, and special event print options for times of eclipse and nodal crossing.

Due to the many options available to the user and numerous high-fidelity models, TRACE is a complex tool which requires time to learn before proper use. For the orbit propagation performed in this report, exposure to the user of TRACE options was limited not only as a step toward demonstrating a proof of concept, but also to show how containerization can be used to make a complicated tool easily accessible.

3. DEX: THE DATA EXPLOITATION PLATORM

In our system, multiple instances of the data propagation capability in TRACE run on the cloud-based Data Exploitation (DEX) platform. DEX is part of the Enterprise Ground Service (EGS) being developed to provide shared cloud infrastructure, enterprise services and common user interfaces for SMC space systems. DEX is a reference architecture and prototype implementation for an evolvable, responsive, and cloud-based data exploitation platform that enables the exploitation of all available and relevant data sources to run a range of data analytics and identify abnormal behaviors in real-time. The DEX data analytics platform provides a software layer between data analytics applications and the underlying cloud infrastructure.

DEX supports fast and easy integration of new and legacy analytics applications and can evolve readily with emergent technologies and products. The DEX platform is designed to store and stream heterogeneous data from diverse resources (satellites, antennae, data centers, etc.); extract information by running configurable, general-purpose and domain-specific analytics; and deliver context-rich and actionable real-time information for satellite operations. DEX also provides intuitive user interfaces that allow users to configure and deploy analytics workflows easily and respond quickly to real-time events by changing the situational, regional, or global analytics workflows running on the DEX platform. DEX's higher fidelity and context-rich characterization of satellite behaviors enable the necessary capabilities for automation of satellite operations in a contested and congested space environment.

3.1 DEX Functional Elements

The main functional elements of the DEX platform are illustrated in Figure 1.

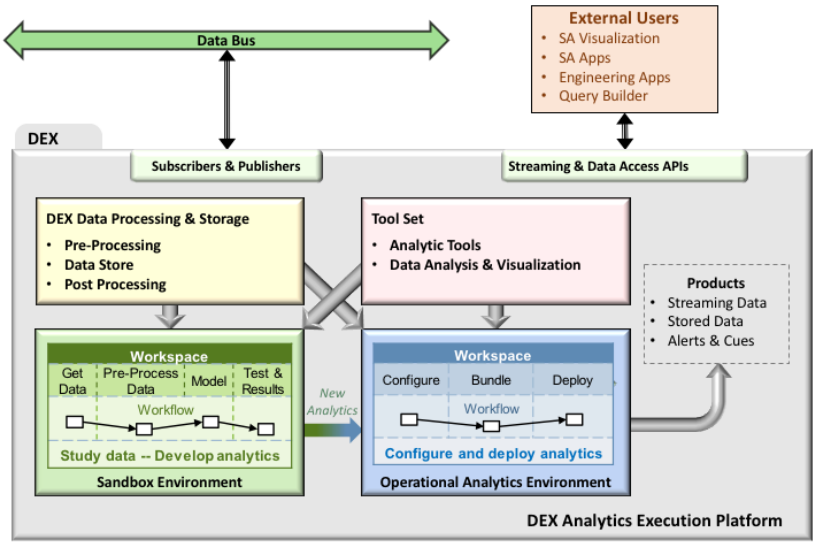

Figure 1: Shows DEX functional elements

These elements include current features of the platform. In addition to the capabilities of the cloud on which it is deployed, the DEX platform provides Data Processing and Storage, an Analytics Tool Set, an Operational Analytics Environment and a Sandbox Environment.

Cloud Capabilities: The DEX analytics platform is deployed on public or private cloud infrastructure. DEX is a cloud-native platform, meaning that DEX components and services were designed to run on the cloud and to take advantage of cloud capabilities, such as autoscaling, elasticity and high availability. DEX supports deployment of applications in software containers to facilitate the easy integration of both new analytics applications and legacy applications. The DEX platform supports resilient operation in the cloud, such as providing replicated deployment of applications. In addition, DEX includes support for software Development and Operations (DevOps) including automated configuration, deployment and monitoring of components.

DEX Data Processing and Storage: DEX includes Data Storage and Processing capabilities for ingested data, including data storage in a schemaless or NoSQL database [7], file-based storage, and data streaming capabilities. Data from different programs may be stored in their native formats in a schemaless database for agility, simplicity and evolvability; schema are applied when data are read. The DEX File Manager allows analytics to read or write data files. DEX support for real-time data streaming allows DEX users or components to subscribe to topics of a data stream, publish alerts and events to a stream, and replay a stream of past data from a specified time.

Analytics Tool Set: The DEX platform provides an expandable Analytics Tool Set that contains tools developed by analytics developers. Since DEX is meant to be a general platform for data analytics, one key goal of DEX is to provide convenient mechanisms for adding new and legacy analytics to the available Tool Set.

The Analytics Tool Set contains both generic and domain-specific analytics tools. These analytics tools may run on different timescales (real time, micro-batch and batch processing). For demonstration purposes, the DEX team has provided an initial set of data analytics, such as Satellite as a Sensor (SAS) neural network based abnormality detection [8-10], Geo-Spatial clustering, Exponential Weighted Moving Average (EWMA) [11], Threshold-based abnormality detection, and Bayesian trend change detection analytics.

In addition to these general-purpose analytics, DEX developers and subject matter experts have integrated applications that address specific program or Space Domain Awareness (SDA) needs. These include:

- A Simplified General Perturbations (SGP4) [12] service that ingests Two Line Element Sets (TLEs), propagates the TLEs, and computes satellite latitude, longitude, and altitude from the propagated TLEs

- The Space Incident Flagging Technique (SIFT), "a method used to monitor the satellite ephemerides to detect changes that are out of the ordinary compared to the object's historical behavior" [13]
- TRACE, the Aerospace Corporation's Trajectory Analysis and Orbit Determination Program that provides high-fidelity orbit propagation, orbit determination, and covariance analysis [3]
- DEX analytics that query data from the Unified Data Library [2]

Operational Analytics Environment: Analytics workflows (groupings of analytics applications) are executed in the DEX Operational Analytics Environment. DEX provides a User Interface that allows users to create and configure workflows of pre-approved analytics from the Analytics Tool Set and deploy those workflows for execution in the DEX Operational Analytics Environment. Analytics workflows deployed on the DEX platform utilize stored and streaming data from the Data Storage and Processing components. Configuration parameters for analytics workflow execution include execution start times, data input and output parameters, and other required analytics parameters. The User Interface allows users to view and manage the lifecycle of workflows running in the Operational Analytics Environment, including starting, stopping, editing or deleting existing workflows.

Sandbox Analysis Environment: Another important component of DEX is the Sandbox Environment. The Sandbox enables users to study DEX data sets by utilizing existing analytics tools from the Analytics Toolset or, when needed, to develop new analytics applications or integrate legacy applications into the Sandbox. Analytics applications running in the Sandbox may utilize stored or streaming DEX data sets. The Sandbox is isolated from the DEX Operational Analytics Environment so that analyses running in the Sandbox do not impact the data or the behaviors of operational system.

The DEX Sandbox allows developers and subject matter experts to develop, configure and validate new analytics tools, including constructing analytics workflows to address complex analytical questions. When a newly created analytics tool has been validated and is sufficiently mature, it will go through an approval process before it is added to the Analytics Tool Set and made available to users of the Operational Analytics Environment through the DEX User Interface.

DEX Data Products: DEX produces data products for both EGS and external users. These data products can be accessed via two Application Programming Interfaces (APIs) for stored and streaming data. The DEX Access API allows users to access stored DEX data. The DEX Streaming API allows users to subscribe to data streams, such as Global Positioning System (GPS) telemetry data or DEX-produced abnormality events. These APIs can be accessed within the DEX Operational Analytics Environment, the DEX Sandbox Analysis Environment, and by DEX external users. In addition to data products, DEX produces Alerts based on analytics-detected events to provide Situational Awareness for EGS users.

3.2 DEX Implementation

Fig. 2 shows the layered DEX architecture and includes the components that are included in the current implementation of DEX. The logos shown on the right correspond to the commercial and open source products used in DEX. Other components were developed by the DEX team.

The lowest level of the layered architecture is the DEX Cloud Deployment and Infrastructure Layer, which consists of cloud virtual machines (e.g., VMWare), Docker containers and Kubernetes container management.

The next layer is the Data Store Layer, which includes open source components including the MongoDB NoSQL database, the CouchDB database, the Kafka data streaming service, the Zookeeper distributed coordination service for distributed applications, and the *etcd* distributed key value store for cluster coordination and state management [14].

The DEX Infrastructure Layer uses the GMSEC (Goddard Mission Services Evolution Center) [15, 16] and UCI (Universal Command and Control Interface) [17] message buses. This layer also includes the DEX-developed Data Streaming API. In addition, this layer includes open source data analytics frameworks such as Spark [18] and Hadoop [19] as well as the Zeppelin web-based notebook [20] and Apache NiFi [21] to automate the flow of data among software components.

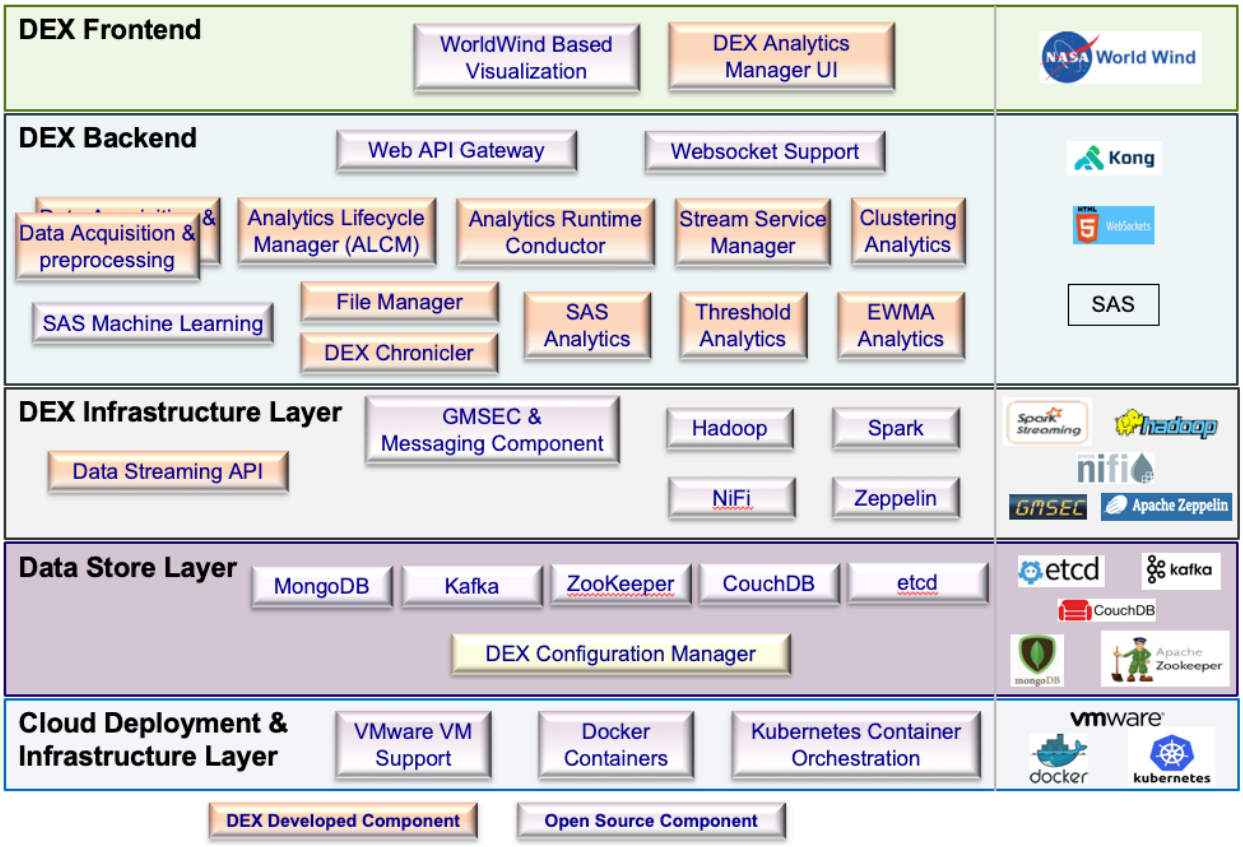

Fig. 2: DEX Implementation Layers

The DEX Back-end Layer contains a number of DEX-developed components, including the Analytics Life Cycle Manager and the Analytics Runtime Conductor that manage the execution of data analytics on DEX, the Data Acquisition and Pre-processing components, and the Stream Service Manager. This layer also contains DEX-developed analytics, including the Threshold analytic that identifies measurands that exceed a specified threshold, the Exponential Weighted Moving Average analytic [11], and a Spatial Clustering analytic application. In addition, the Back-end Layer includes the commercial SAS neural network application for detecting data abnormalities, which DEX developers containerized for deployment on the cloud. The containerized TRACE and UDL data access analytics (not shown) are also in this layer. This layer also contains open source Web API Gateway and Websocket components for exchanging data outside of DEX.

Finally, the DEX Front-end Layer includes the DEX-Developed Analytics Manager User Interface as well as the Open Source WorldWind-based visualization [22] that is used to display the results of DEX analytics.

4. LEOLABS DATA AND THE UNIFIED DATA LIBRARY

Our work utilizes state vectors generated by LeoLabs [1], a company that operates multiple radar stations around the globe and tracks thousands of objects in low earth orbit (LEO). These radar instruments include the Poker Flat Incoherent Scatter Radar (PFISR) near Fairbanks, Alaska, the Midland Space Radar (MSR) near Midland, Texas, and the Kiwi Space Radar (KSR) located in the Central Otago region of New Zealand. LeoLabs data includes observation data, state vectors and conjunctions for satellites, debris, and other low earth orbit objects. These data are published to the Unified Data Library. Our experiments utilize LeoLabs state vectors for satellites.

The UDL [2] is a repository for Space Domain Awareness (SDA) data from many sources. The UDL is being developed by BlueStaq for the AFRL, Air Force SMC and Air Force Space Command (AFSPC). Data sets from

different providers on the UDL are normalized using a common model representation. The UDL provides several interfaces for accessing data, including a REST (Representational State Transfer) API that provides access to over 40 data sources; a History REST API that provides access to 12 services that serve historical UDL time series data (e.g., TLEs, Observations, Maneuvers, and State Vectors); a Bulk Data Access API that provides an asynchronous mechanism to request large data sets for analysis and machine learning; and a near real-time data streaming capability using UDL Secure Messaging that is based on Apache Kafka. The UDL data download application uses the REST API to query LeoLabs data in the UDL; these state vectors are then analyzed by the TRACE application.

5. INTEGRATION OF TRACE, DEX AND UDL

To enable deployment of multiple instances of the TRACE application on modern cloud computing infrastructure, we packaged the TRACE software with all its dependencies. We also packaged a separate application that downloads data from the UDL. Once these applications were packaged for cloud deployment, we configured and deployed an analytics workflow on the DEX platform that includes the UDL data download application as well as multiple instances of the TRACE application, as shown in Fig. 3.

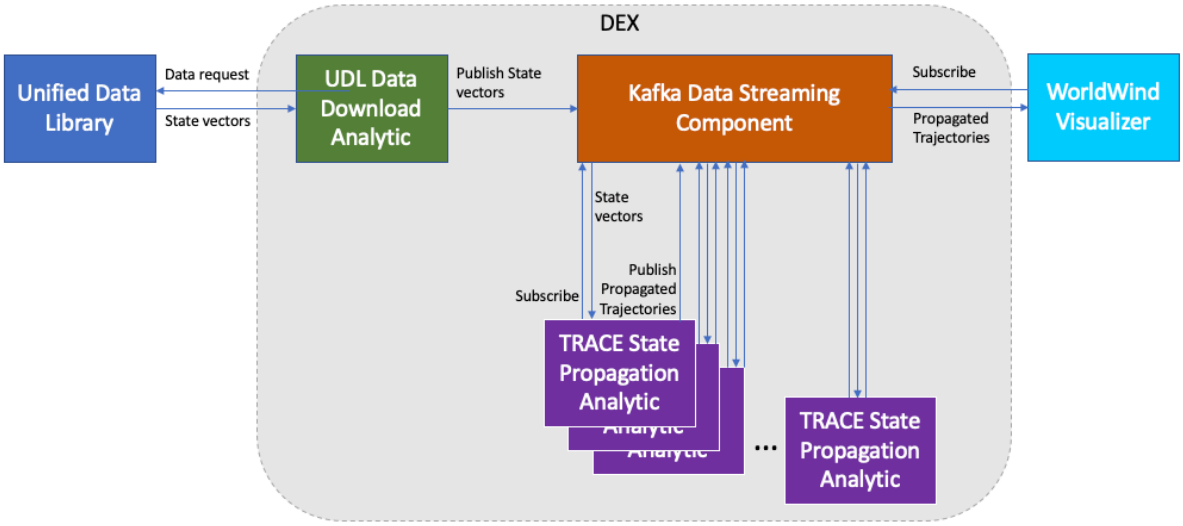

Fig. 3: Shows the integration of UDL, the data download analytic, multiple instances of the TRACE state propagation analytic, and the WorldWind visualizer.

The first analytic application in the workflow downloads LeoLabs state vectors from the UDL and publishes the state vectors to the Kafka data streaming platform, which also runs on DEX. Multiple instances of the TRACE analytic application running on the DEX platform then subscribe to the Kafka topic to retrieve state vectors and run the TRACE high fidelity orbit propagation software on those state vectors. The experiment documented in this paper used ten instances of the TRACE analytic. The results of each orbit propagation are then published to a Kafka data stream, where they may be further analyzed or visualized by other components running on the DEX platform. In our workflow, we visualize the propagated orbits using the NASA WorldWind [22] visualization tool.

In the sections that follow, we discuss key aspects of the integration of the TRACE analytics, downloads from UDL and the DEX platform.

5.1 Virtualization of Analytics Using Software Containers

Converting a legacy application like TRACE to run on a modern cloud infrastructure can be challenging. The easiest approach is to use a virtualization technique called *containerization* that packages the legacy application and all its dependencies in a single software container that can be easily deployed on the cloud.

Virtualization techniques allow sharing of physical resources (processors, memory, networks, etc.) by multiple applications. Virtualization techniques such as *virtual machines* and *containers* are widely used in cloud computing

to allow many applications to share the resources of the physical machines in a data center. A virtualized resource presents each application or service with a model of computation, storage and communication. To enable different workloads to be co-located on nodes in a cloud computing environment, virtualization technologies must support the *isolation of virtualized workloads* so that different workloads running on the same physical node don't affect one another's performance or correctness and provide *resource management* to control the resources consumed by each workload. Fig. 4 shows the two most common virtualization types. Virtual machines (VMs) each receive a slice of the underlying server hardware, with hypervisor software managing multiple VMs. Each VM contains a full guest operating system (OS) and the application that runs on that OS. By contrast, containers are a more lightweight alternative to VMs that are faster to deploy. Each container shares a slice of a single operating system kernel, so containers are smaller than VMs. Applications are packaged in containers with all of their software dependencies. Both containers and VMs are portable and can be deployed easily on a cloud, but VMs are more heavyweight and may take minutes to deploy, compared to seconds to deploy a container. Containers require fewer resources than VMs and can be deployed more densely on cloud resources, resulting in improved resource utilization and lower power usage of cloud resources. As a result, for large scale application deployments that need to scale quickly with user demand, containers have become widely used in industry for cloud deployments.

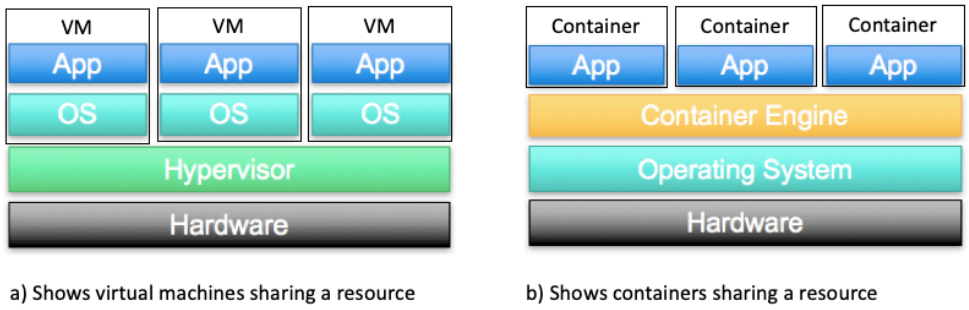

Fig. 4: Virtualization comparison of Virtual Machines and Containers

The DEX platform deploys applications as containers. To run a legacy application like TRACE on the DEX platform, the application must first be "containerized" or packaged as a software container.

5.2 TRACE Application Containerization

The TRACE team containerized the application by packaging the TRACE software along with all of its dependencies in a Docker container. Docker [23] is the industry leader and the de facto standard for container technologies.

The containerized TRACE analytic used in this project only exposes the specific TRACE functionality related to orbit propagation. For this project, we expose only a portion of the overall TRACE capabilities to provide focused and simplified use of a complex engineering tool. The other capabilities of TRACE are made unavailable, and user access to TRACE is controlled and limited. In the future, we plan to develop additional DEX analytics that expose other aspects of TRACE functionality.

In particular, the TRACE analytic requires only a simplified set of parameters to reduce the complexity for the user. Typical TRACE propagation input scripts allow the user to adjust over 40 parameters. Their impact ranges from selecting and describing dynamic models to specifying output formats and units. The containerized version of the application significantly reduces this complexity. The only inputs required are the output interval, stop time, coordinate system, and gravity model order, as well as the state vectors to be propagated that are downloaded from the Unified Data Library. The reduction in the number of required inputs simplifies the use of the application but maintains access to pre-defined high-fidelity physical models, including drag, solar radiation pressure, and third body forces, which are all included using pre-selected models.

Fig. 5 shows the containerized TRACE application. The software container includes all the dependencies that are required to run the TRACE analytic. This includes six databases that include earth orientation parameters (EOP), leap seconds table, planetary ephemerides, gravity models, etc. These databases are described in detail in Table 1.

The software container also includes a software wrapper written in Python that makes the TRACE analytic long-running and persistent, so that the orbit propagation functionality runs whenever the TRACE analytic receives a new state vector on its input stream. The wrapper produces the required input file for the TRACE code for each state vector that it receives. Once the TRACE software ingests the state vector, propagates the satellite trajectory, and produces an output file, a separate parser written in Python processes the output file and produces the desired output format, which is either a software file stored in the DEX File Manager or an output data stream published to Apache Kafka on DEX.

These components are packaged as a Docker container, and the resulting Docker image is uploaded to the DEX environment along with an analytic manifest file that describes the required inputs and parameters for the TRACE analytic. Once uploaded, the TRACE analytic becomes available for selection and configuration by DEX users via the DEX User Interface.

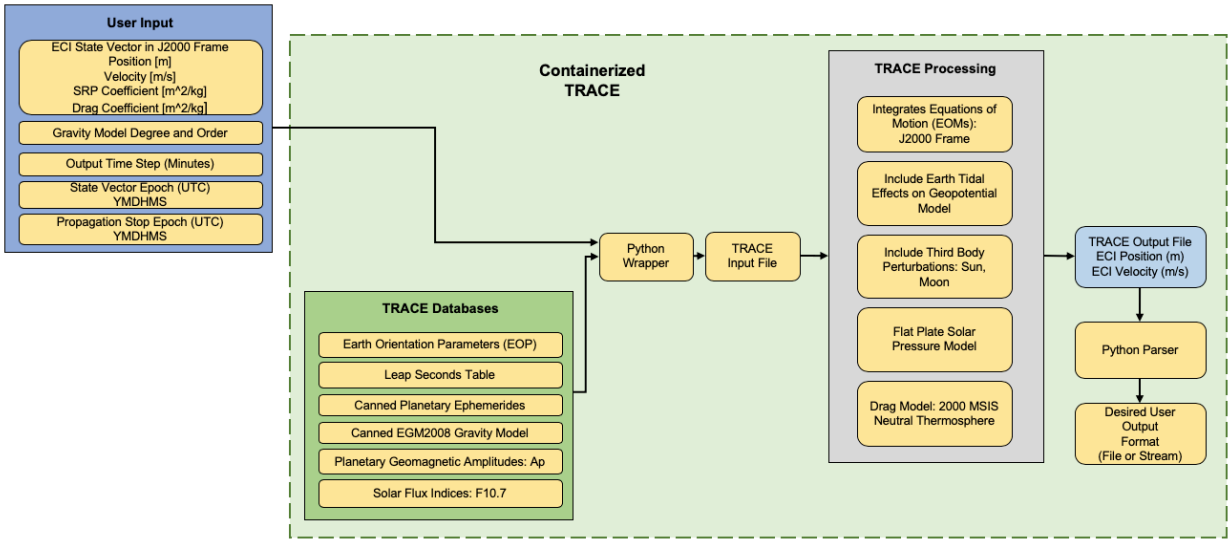

Fig. 5: Shows the components that are packaged together in the TRACE software container. These include TRACE processing modules, six databases that are used in running TRACE, a python wrapper, python parser, TRACE input and output files, and a module that produces the desired output from the analytic as either a file or data stream.

Table 1: Shows the TRACE support files and databases that are packaged in the TRACE software container.

TRACE Support Databases and Files	***Description***
Earth Orientation Parameters	File containing the IERS Earth Orientation Parameters that describe the irregularities of the Earth's rotation. The parameters are utilized for coordinate frame transformations between Earth-Centered Inertial (ECI) and Earth-Centered, Earth-Fixed (ECEF)
Leap Seconds Table	Table of the time difference International Atomic Time (TAI) – Coordinated Universal Time (UTC)
Canned Planetary Ephemerides	x86 Jet Propulsion Laboratory (JPL) Development Ephemeris (DE) 200 file containing planetary ephemerides. It is utilized for inclusion of 3rd body gravitational effects
Canned EGM2008 Gravity Model	The Earth Gravitational Model EGM2008 consisting of spherical harmonic coefficients published by the National Geospatial-Intelligence Agency (NGA)
Planetary Geomagnetic Amplitudes: Ap	File containing a table of daily planetary geomagnetic amplitudes Ap. The Ap values are utilized by the 2000 Mass-Spectrometer-Incoherent-Scatter (MSIS) atmosphere model in TRACE
Solar Flux Indices: F10.7	File containing a table of daily solar flux F10.7 indices. The F10.7 values are utilized by the 2000 MSIS atmosphere model in TRACE

Table 2: Shows the input values that must be specified for a TRACE analytic instance in DEX.

Input Value	*Description*
Stop Epoch	Final epoch time to integrate satellite trajectory to in UTC: Year, Month, Day, Hour, Minute, Seconds
Time Step	Trajectory output rate in minutes
Gravity Model Degree and Order	Degree and order of the harmonic coefficients to include in the gravitational model (e.g. 70x70)
State Input Coordinate Frame	Cartesian or Keplerian
Group_ID	Specified when multiple instances of the TRACE analytic are consumers of state vectors on the same Kafka stream
Partition_Assignment	Used if an instance of TRACE is assigned to use a specific partition of a Kafka stream

Table 2 shows the input values that are specified when configuring an instance of the TRACE analytic via the DEX User Interface. Fig. 6 shows an example configuration. In addition to the parameters specified in Table 2, the user also specifies the data input, which is a Kafka data stream to which the TRACE analytic subscribes. When the separate UDL data download analytic publishes LeoLabs state vectors from the Unified Data Library to the Kafka stream, the TRACE analytic then processes the state vectors by running the TRACE code. The TRACE analytic configuration also specifies the data output, which in Fig. 6 shows that the output is a Kafka stream where the TRACE analytic publishes the propagated trajectories generated by TRACE.

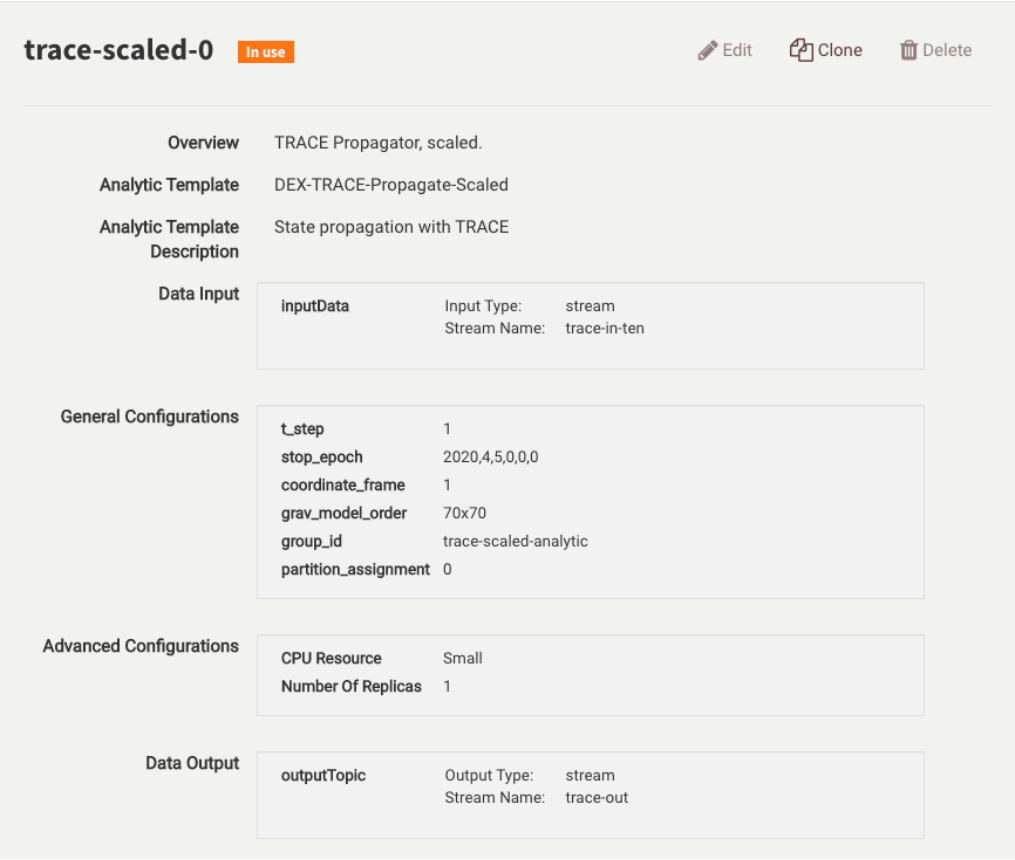

Fig. 6: This screenshot from the DEX User Interface shows the configuration of an instance of the TRACE analytic.

5.3 UDL Data Download Analytic Containerization

An application that downloads LeoLabs state vectors from the UDL was also containerized to run on the DEX platform. The application is called the UDL-LL-scaled analytic, where "UDL" refers to the Unified Data Library,

"LL" indicates the data being downloaded from the UDL are LeoLabs state vectors, and "scaled" refers to the fact that that analytic can fetch state vectors from multiple satellites during a specified epoch. The parameters for configuring the analytic are shown in Fig. 7. These parameters include a token used to access the UDL (obscured), a list of one or more NORAD IDs (North American Aerospace Defense catalog numbers) corresponding to satellite objects whose state vectors will be downloaded, the epoch of interest, and the number of partitions in the Kafka data streaming platform that the application uses to publish state vectors. The analytic configuration also includes the name of an output stream.

When the UDL-LL-scaled analytic instance is deployed on DEX, it uses these parameters to query the Unified Data Library for state vectors corresponding to the NORAD IDs and the specified epoch. When the UDL query returns one or more state vectors, the UDL analytic publishes these to one or more partitions of a Kafka output stream. If the input parameter specifies multiple partitions, the application performs load balancing by publishing the state vectors for different satellite objects to different Kafka partitions.

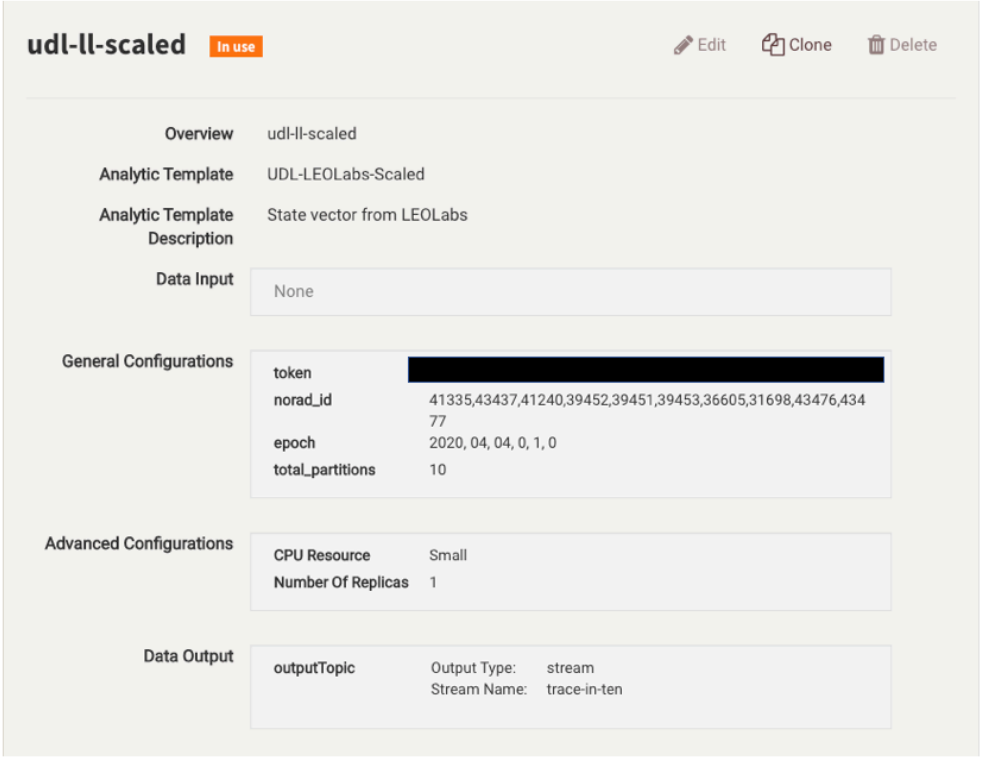

Fig. 7: Shows a screenshot for the DEX User Interface with an example of a configuration of the UDL data download analytic.

The current implementation of the UDL-LL-scaled analytic queries the state vectors once and then completes its execution. Alternatively, we could implement a long-running analytic that continues to query state vectors for subsequent epochs.

5.4 NASA WorldWind Visualizer

To visualize the orbit trajectories generated by the TRACE applications, we use the NASA WorldWind visualization software [22]. NASA WorldWind is an open source virtual globe API and Software Development Kit (SDK) that allow developers to create interactive visualizations of 3D globe, map and geographical information.

The WorldWind application is deployed on the same cluster as the DEX platform but is separate from DEX. WorldWind subscribes to the Kafka topic to which the propagated trajectories are published by the TRACE analytics running on DEX. WorldWind then visualizes these trajectories on a globe.

5.5 Creating a Bundle or Workflow of Analytics

A DEX ***bundle*** is a grouping of one or more analytic instances that are intended to be deployed and executed together as a workflow on the cloud infrastructure. For DEX, the bundle is the unit of instantiation and execution on the DEX platform. Even if a user wants to deploy a single analytic application, the user must create a bundle that contains that analytic to deploy it on the DEX platform.

Fig. 8 shows the two analytics created for our experiments. The first bundle, *udl-ll-scaled-bundle*, contains a single instance of the udl-ll-scaled analytic that downloads Leolabs state vectors from the Unified Data Library. The second bundle, *trace-scaled-bundle*, contains ten instances of the trace-scaled analytic that run in parallel and run the TRACE orbit trajectory software on the downloaded state vectors.

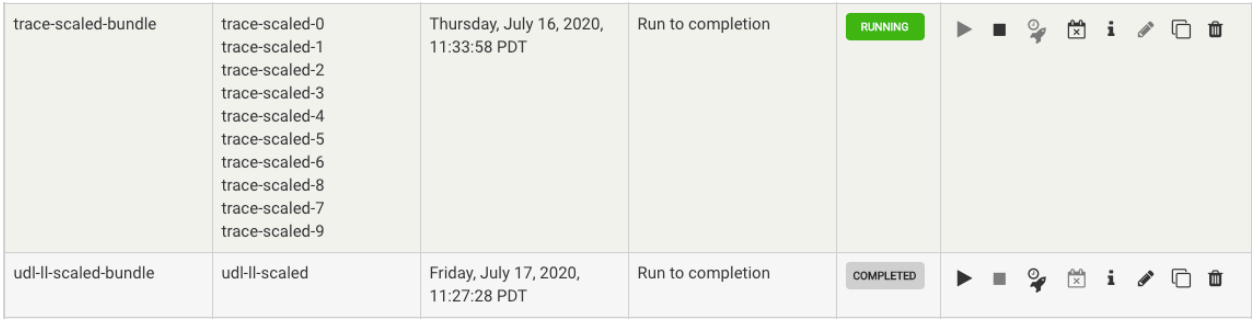

Fig. 8: Shows a screenshot from the DEX User Interface of the two bundles, one bundle that downloads data from the UDL and a second bundle than contains ten instances of the TRACE analytic running in parallel on the cloud.

5.6 Deployment of UDL-LL and TRACE Analytics on DEX

Fig. 9. shows the deployed bundles for our experiment on the DEX platform, including the UDL-LL-scaled-bundle that downloads LeoLabs state vectors from the UDL, and the TRACE-scaled-bundle that includes ten instances of the TRACE analytic to propagate state vectors in parallel. The projected trajectories are visualized using the NASA WorldWind software. We successfully executed these bundles on the DEX cloud platform and visualized the results, as shown in the next section

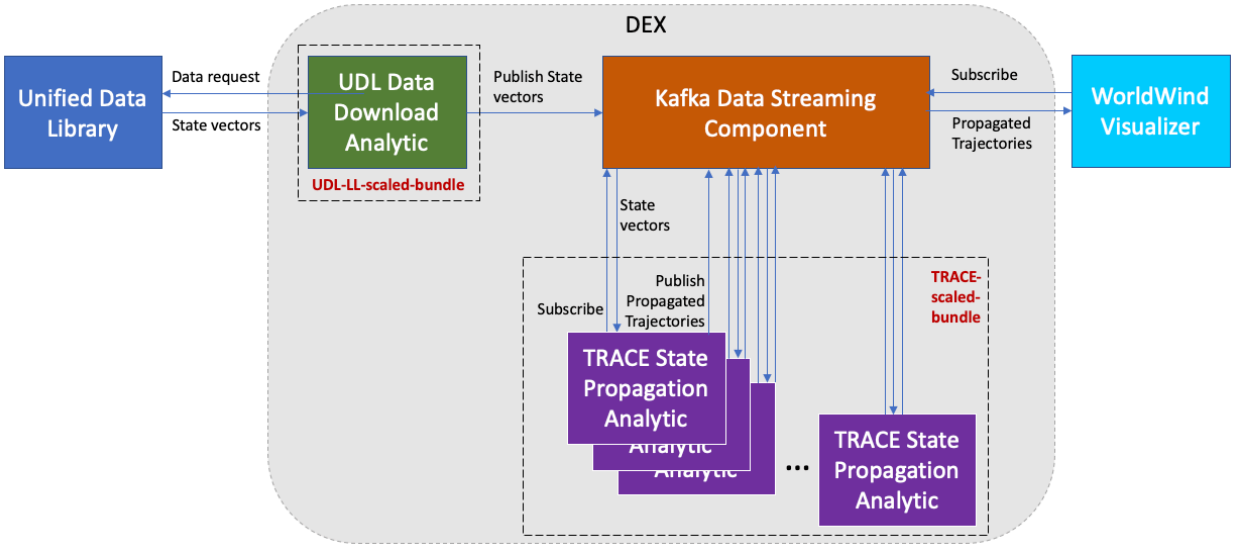

Fig. 9: Shows the deployed bundles on the DEX platform

5.7 WorldWind Visualization

Fig. 10 shows the NASA WorldWind visualization of the propagated orbit trajectories for the 10 LEO satellites whose state vectors were downloaded from the UDL. These orbit trajectories are labeled by their NORAD IDs.

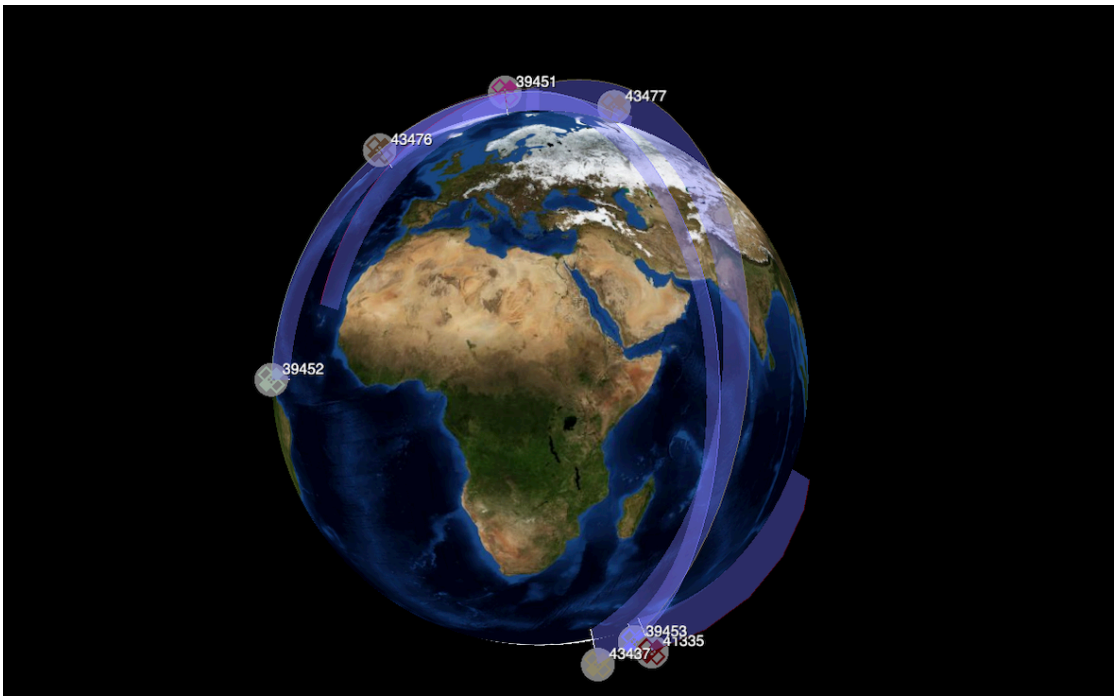

Fig. 10 shows the propagated orbit trajectories for 10 satellites analyzed in parallel on the DEX cloud platform.

6. SUMMARY AND FUTURE WORK

The goal of this work was to demonstrate that cloud computing can support the automated analysis of a large number of objects in congested and contested space. As the number of space objects rapidly increases, the ability to perform automated, high fidelity orbit propagation on many of these objects in parallel is urgently needed. Adding more Satellite Vehicles (SVs) for orbit propagation should be supported easily by modifying the configuration of the associated analytics. This allows for on-demand and fast insertion of tracking capabilities for newly introduced SVs.

We integrated the Aerospace Corporation's TRACE Trajectory Analysis tool with the cloud-based DEX analytic platform by packaging the TRACE software in a container for easy deployment on the cloud. We also developed and containerized an application that queries state vector data in the Unified Data Library; this application downloads state vector data generated by LeoLabs for objects in low earth orbit. The analytics workflow deployed on the DEX platform propagates the downloaded state vectors in parallel by running multiple instances of the TRACE software on a cloud. The resulting propagated trajectories are then visualized using the NASA WorldWind visualization software.

For our experiments, we deployed ten instances of the TRACE analytic application on the cloud along with a single instance of the UDL application that downloaded state vector data from the UDL for 10 different satellites. The techniques we demonstrated can be used to scale both the number of TRACE instances and the number of UDL data download application instances to arbitrarily larger numbers to analyze more objects.

In future work, we will continue to scale the number of objects for which we download and analyze data. In addition, the current TRACE orbit propagation analytic exposes only one part of the functionality of the TRACE software. Going forward, we will create additional analytics that containerize and expose additional TRACE capabilities for use by operators and analysts.

The DEX team is also containerizing and integrating additional Space Domain Awareness applications, including an SGP4 service, the Space Incident Flagging Technique (SIFT), and others. By containerizing the legacy TRACE application, which has been developed over five decades, to run on a modern cloud platform, we have demonstrated the potential to containerize, deploy and scale other key legacy SDA applications on cloud infrastructures.

7. REFERENCES

[1] "LeoLabs." https://www.leolabs.space/ (accessed 2020).
[2] "Unified Data Library." https://unifieddatalibrary.com/ (accessed 2020).
[3] C. C. Tonies, "TRACE Orbit Determination Program Version D," AEROSPACE CORP EL SEGUNDO CA EL SEGUNDO TECHNICAL OPERATIONS, https://apps.dtic.mil/dtic/tr/fulltext/u2/489436.pdf, 1966.
[4] A. L. Chervenak *et al.*, "The Data Exploitation (DEX) Platform," presented at the 35th Space Symposium, Technical Track, Colorado Springs, CO, USA, April 8, 2019.
[5] J. Y. Cruz, M. D. Menn, W. A. Feess, and V. Nuth, "GPS Constellation Navigation: L-Band Measurements Only," in *Proceedings of the 2005 National Technical Meeting of The Institute of Navigation*, 2005, pp. 792-798.
[6] R. H. Prislin, D. C. Walker, and R. J. Mercer, "TRACE66 Trajectory Analysis and Orbit Determination Program. Volume 1. General Program Objectives, Description, and Summary," The Aerospace Corporation, 1971.
[7] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Record,* vol. 39, no. 4, pp. 12-27, 2011.
[8] C. Bowman, G. Haith, C. Tschan, and P. Zetocha, "The Search for Signatures of Space Weather Effects," presented at the AIAA Infotech@ Aerospace, AIAA SciTech Forum, San Diego, CA, USA, 2016.
[9] M. Hammond and R. Jobman, "Satellite-As-a-Sensor Neural Network Abnormality Classification Optimization," in *Small Satellite Conference*, https://digitalcommons.usu.edu/smallsat/2006/All2006/16/, 2006.
[10] "Logical Designs Consulting." http://www.logicaldesigns.com/Index.htm (accessed 2019).
[11] *EWMA Control Charts, Engineering Statistics Handbook*, National Institute of Standards and Technology, 2013. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section3/pmc324.htm
[12] D. Vallado and P. Crawford, "SGP4 orbit determination," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2008: https://arc.aiaa.org/doi/pdf/10.2514/6.2008-6770, p. 6770.
[13] R. Swartz, J. Coggi, and J. McNeill, "A swift SIFT for satellite event detection," in *AIAA/AAS Astrodynamics Specialist Conference*, 2010, p. 7527.
[14] "etcd." https://etcd.io (accessed 2020).
[15] R. Detter. "Goddard Mission Services Evolution Center (GMSEC) Overview Presentation." NASA Goddard Space Flight Center. https://gmsec.gsfc.nasa.gov/referenceMaterials.php (accessed 2019).
[16] "GMSEC Technical Overview," *Goddard Tech Transfer News,* vol. 9, no. 2, Spring 2011. [Online]. Available: https://gmsec.gsfc.nasa.gov/images2/TechTransferNews-Spring11.pdf.
[17] US Air Force Virtual Distributed Laboratory (VDL). "Universal Command and Control Interface (UCI)." https://www.vdl.afrl.af.mil/programs/oam/uci.php (accessed 2020).
[18] "Apache Spark." https://spark.apache.org/ (accessed 2019).
[19] "Apache Hadoop." https://hadoop.apache.org. https://hadoop.apache.org (accessed 2019).
[20] "Apache Zeppelin." https://zeppelin.apache.org/ (accessed 2019).
[21] "Apache NiFi." https://nifi.apache.org/ (accessed 2020).
[22] "NASA WorldWind." https://worldwind.arc.nasa.gov/ (accessed 2020).
[23] "Docker." https://www.docker.com/ (accessed 2019).