Part 1: Table of Contents

Part 2: Identification and Significance of the Innovation

Summary of the Innovation

We propose to create a new storage, processing and retrieval schema for scientific data, with an emphasis on planetary remote sensing, that combines the power of cloud computing, relational databases, and thorough metadata to dramatically improve the efficiency, usability and scientific return on investment of spacecraft missions. The working title for this project, where planetary spacecraft meet the cloud, is “Aerobrake.” Aerobrake will allow users to search, manipulate, and simultaneously use multiple planetary data sets within a cloud storage and processing system. It will include the option to download subsets of stored data to a local machine, either as standalone files or within an instance of an interpreted programming language (such as IDL, Python, or Matlab). To minimize redundant processing, the database will permanently store Derived Data Layers (DDLs) which might include calibrations, projections or model fits. It will also retain all relationships between raw, derived, and state data. This rich, metadata-driven relationality will make all derivations completely lossless and reversible and allow users to perform complex cuts, queries, and manipulations on data across DDLs.

Planetary Data Archival in the Big Data Era

Scientific institutions in general lag far behind the industry state of the art in data management. Aerobrake is an attempt at modernizing archival and access functions historically provided by NASA’s Planetary Data System (PDS) [1]. The PDS does an important and admirable job of serving data to the planetary science community. However, born out of a storage-centered mandate nearly three decades old, it has not adapted well to the "big data" era.

In other words, planetary data access technologies have not grown in pace with the volume of the data archive, which has expanded dramatically in the last decade, and is expected to surpass 1 petabyte sometime in the next few years [2]. Scientists who wish to use this data face a variety of barriers that make their research slow, difficult to budget, and sometimes entirely impossible. These barriers include both search costs (difficulties in simply figuring out whether or not the data a researcher is interested in exists at all, and if so, where in the corpus it exists) and costs associated with additional post-processing of data retrieved from the PDS archive, which is typically served in raw or minimally-reduced formats.

These barriers are created in large part by how the PDS stores and serves data. The

PDS's methods, which date from the compact disk era [1], are designed for the convenience of data archivists rather than the end-users of data products. They require end-users to engage in complex, inefficient methods to access and analyze PDS data. Although the recent upgrade to PDS4 has made many search operations easier by adding XML headers to PDS files and reorganizing data into "collections" of similar types of data products [3], PDS has not otherwise significantly changed the way that end-users access PDS data. In particular, it has not harnessed the potential of recent advances in cloud-based storage and processing to improve how data is searched, served, or used.

In order to even determine which PDS files might be relevant to their work, researchers must use some metadata search tool—Java Mission-planning and Analysis for Remote Sensing (JMARS) [4] is the most popular, but Google Mars, the PDS Orbital Data Explorer (ODE) [5], and the USGS Planetary Image LOcator Tool [6] are also widely used—to determine which PDS files might be of interest. The metadata tags by which these files are organized are documented, but inconsistently used, and new tags are often defined according to the exigencies of the instrument or mission rather than subsequent science needs. JMARS and similar geospatial information systems partly patch this problem by associating each supported mission's metadata with regions on planetary surfaces, allowing researchers to define latitude and longitude ranges and receive pointers to associated PDS files. However, the researcher must still download these files locally to investigate them, and these files might be terabytes in size. Thus, simply downloading data of interest may take hours or days and requires researchers to possess significant digital storage capacity.

While the PDS has some standards for archived data and technically contains all of the information required for understanding raw data and the context in which it was collected, formats vary widely, cannot be processed server-side, and often require additional tools to read into standard planetary data processing languages and environments. Thus, in order to actually investigate these files, the researcher must process local copies (according to calibrations, map projections, mosaics or analytical products) into a usable format which requires additional local storage and processing power. Some missions archive calibrated (so called "reduced") or map-projected data products in the PDS, but such archival products "lock in" the specific calibration and projections favored by the instrument team, which may not be useful or acceptable to individual researchers. There are some notable attempts at making tools for processing PDS data, including the Integrated Software for Imagers and Spectrometers (ISIS) software packages curated by the USGS Astrogeology Science Center [7], the CRISM Analysis Toolkit (which utilizes ENVI and IDL) [8], and the OMEGA Reduction Software (which utilizes IDL) [9]. All such packages (and their underlying image processing or data visualization software) have steep learning curves, which add to the time and effort necessary for a researcher to work with planetary data. Also, many of these tools require proprietary software, which may be expensive for individual researchers without access to institutional licenses. Moreover, such tools do not even exist for all data sets, and so researchers are often required to create their own or do without.

This process is inefficient. At best, it incentivizes the construction of redundant computing infrastructures, the design and implementation of similar software products, and the wasteful duplication of research efforts between teams. At worst, it leads to the termination of promising projects due to technical, data management, and processing barriers.

All of these problems, of course, compound when researchers wish to co-analyze datasets from separate instruments or missions. The likelihood of finding usable existing tools decreases greatly; it's not unusual for a researcher to find that the data from each of the instruments they're examining in a combined study was differently calibrated and stored in mutually incompatible map projections. For instance, even though these instruments are on the same spacecraft, CRISM imaging data are usually map-projected using the 128-pixel-per-degree Mars Orbiter Laser Altimeter (MOLA) shape model of Mars, while HiRISE image projections typically use a Mars sphere model (which does not account for variations in local shape). A researcher may therefore need to exert a great deal of additional effort and accept large amounts of distortion in the data in order to get multiple datasets to line up in a region of interest. Each distinct data set may each be tens to hundreds of terabytes in size, so even if software tools for processing the data *are* available, doing so requires access to a yet larger computational infrastructure than a single-instrument study requires.

Planetary science is in a situation where technological barriers to entry discourage data use, which ultimately reduces the scientific return on investment of missions.

Enhancing Planetary Data Archival by Maximizing Metadata

Some of these issues are storage and retrieval limitations due to the limitations of existing software. Many of these limitations could be mitigated by leveraging cloud data storage and processing. However, most of these issues are, fundamentally, due to insufficiencies in available metadata.

In the general sense, metadata is data that gives *context* to some other piece of data. Understanding whether specific data is of interest is a metadata search task. This task is impossible when the appropriate metadata is not available. Camera calibration procedures, standards, and software are all metadata. Camera *definitions* necessary to compute the geometry of a map projection from raw data are also a kind of metadata. Part of the problem—a known problem, highlighted in a recommendation by the 2012 Planetary Data Workshop [2]—is that extensive and high-level metadata is not always made available by data providers, or made available in nonstandard or difficult-to-use formats.

Another, less well-acknowledged problem, is that even if all of the previously mentioned types of metadata were available for every data set, we could *still* improve the usefulness and usability of the data set by improving metadata handling and volume. This could be accomplished by directly linking metadata to its associated data in an easily-usable database architecture and dramatically expanding the scope and volume of metadata associated with observations. These improvements would centrally aim to provide *the total available context for every piece of stored data*—with metadata itself considered a category of data. This involves a major increase in the total amount of stored data—but modern cloud computing platforms have the capacity to handle this storage volume without propagating storage costs to end-users.

So how do we maximize the useful metadata contained in a database of remote sensing data—along with the usefulness of this metadata? An object-relational database (or any other relational database) is a kind of metadata management machine; by its very nature, it provides context for entries (such as pointers to other objects). Cost, size, and complexity considerations, rather than research utility and usability considerations, have historically determined the quantity

and quality of the relational metadata provided in databases of remote sensing data.

In a planetary science context, observational metadata has traditionally referred to 1) spacecraft state information (such as orientation, pointing, time, temperature), 2) detector state information (such as exposure time and operating mode), and 3) instrument / camera definitions (such as CAHVOR models, focal lengths, angular resolutions). But the term metadata should also refer to derivational data like algorithms, modeled detector behavior, calibration products, and software. All of these provide context for both raw and derived data. What's more, raw observational data provides context for derived data and derived data provides context for raw, so each is metadata for the other; but our current methods of storing and retrieving such data often "break" such relations and make it impossible to determine the true and complete context of every datum. In other words, all data is metadata, if it is put into context—and that context can be defined quite naturally and efficiently by relational mappings in databases.

If we're willing to deal with the added complexity and storage requirements, we can make all data abstractions lossless and reversible with metadata. To give one very important example, the first step in doing research with raw remote sensing data is always to get it into some kind of array—which is of course a type of data abstraction. This often accounts for a large fraction of the total research effort. It also often requires a rasterization or interpolation step (like mapping elongated "pushbroom" pixels onto a regular grid, co-registering data, or creating a Digital Elevation Model). Rasterization and interpolation are often performed at intermediate processing steps to reformat data as an array, even if end-users are not specifically interested in using the derived data from those intermediate processing steps, and such reformatting comes with a cost in terms of both computational overhead and information loss.

This means that to maximize the efficiency and downstream usability of data, *rasterization and interpolation should be performed as infrequently and as late in data processing as possible*. A database that retains complete metadata for every data abstraction makes all rasterizations and interpolations lossless—all its rasterized/interpolated data contains metadata about how and from where its values were derived. Moreover, if the database retains complete metadata for *all of the individual components of every observation and abstraction*, it becomes relatively easy to perform many common secondary operations on raw data, such as calibration and map projection, without having to rasterize the data at all.

Putting Planetary Data in the Cloud

Thus, we propose Aerobrake, a cloud-based, integrated storage and processing system for planetary data. By moving storage and processing of planetary data into the cloud and improving its overall intercompatibility, Aerobrake will make planetary science data much more *usable*: reducing difficulty and duplication of work will improve the return on research hours spent. Aerobrake will also make this data much more *useful*: it will increase the total amount of science that researchers can produce from a given family of datasets. Furthermore, by serving as a centralized storage and processing system, Aerobrake will eliminate the need for each researcher to have his or her own large local system for downloading, storing and processing large amounts of data. This will reduce the upfront costs of planetary science projects and increase the total return on research dollars spent.

Our proposed move to cloud based storage rests on the insight that raw remote sensing

spacecraft data can be represented within a database by relational mappings and state descriptions for the smallest meaningful units of data. New ADLs can be built on this layer by simple algorithmic transformations that preserve the origin of the abstracted data.

As a concrete example, we will explain how Aerobrake would approach one common form of remote sensing data: a CCD image. The smallest meaningful unit of a CCD image is a pixel value (usually described as a Data Number or DN, which is a scale factor of the number of photons in the CCD well). Aerobrake, rather than treating a CCD image of a scene as a unit whole, would treat each well within the CCD as its own single-pixel detector. The state information of each of these pixels information would be identical to the state information of the original CCD image as a whole (generally including time, spacecraft position and orientation, camera model, temperature, and operation mode) with a single additional value that defines the pixel extent and its relative position on the detector. The DN would be stored in the Aerobrake database as a single entry, relationally linked to its position value and the appropriate metadata—in this case, the state information for the CCD image as a whole and a set of generic information about the instrument, such as focal length, sensitivity, and bandpass. It's simple to see that this representational data would be equivalent in total memory usage as the (uncompressed) raw data, and also that it would be straightforward to reconstruct the raw image from the individual pixel entries.

In other words, no information is lost by storing the data in this way. However, it makes the data much more *flexible.* Because Aerobrake treats each pixel value as a separate database entry, users will be able to define complex cuts on the data that enable them to acquire only the specific data that is relevant to their project. Similarly, it will be easy for them to map new relationships between specific pixels or regions on the detector and other pixels, regions, or derived values.

Aerobrake will also make many common secondary operations on remote sensing data much easier and more efficient. For instance, consider the common case of projecting an image onto a Digital Elevation Model (DEM). This process always involves application-specific tradeoffs and decisions. The image, which is defined according to the camera projection, does not share the same "sample points" as the DEM, which is defined as a regular grid on the surface. Such projections thus often require interpolation, and the appropriate interpolation algorithm differs between applications. Moreover, different applications prefer entirely different types of projection (as conic, cylindrical, etc). Some methods of projection and interpolation are better at preserving volume information, some at preserving angular relationships between elements, some are more appropriate for specific latitude ranges on a spheroid, and so on. In the projection process, a great deal of information about the origin and other characteristics is almost always lost.

However, if an Aerobrake user wishes to project an image stored in the Aerobrake database onto a DEM, they will not need to do so by immediately creating a lossy raster image. Since Aerobrake will define pixels independently and associate them with camera state (and thus camera pointing / projection information), a DEM projection from the image in the Aerobrake database can simply be defined as a *new set of pixels* (a Derived Data Layer). Pixels in this new set will likely have irregular (and perhaps even multidimensional) shapes. This set will also have its own set of associated metadata related to the projection method, perhaps including a normal vector or a center of mass coordinate. However, *each projected pixel will*

also retain a mapping to its source(s) in the raw data.

If the user requires an interpolated projection, they can then create one from this set of projected pixels. This is one of the major advantages of Aerobrake—because the system will allow data rasterization to be delayed until the final step in image processing, this computationally expensive and lossy step will only be done once. Furthermore, since the interpolation is being performed within Aerobrake, it can *also* be defined as a new set of pixels (or Derived Data Layer). This set can also be lossless, simply by defining the interpolation in such a way that each pixel in the interpolated pixel set also contains information about how its value was obtained—in other words, which pixel values from the DEM-projected pixel set were used to obtain its values and how. At this point, the user would have an interpolated, map-projected raster image in which each pixel could be mapped directly back to the original detector value *and* all associated state and transformation metadata.

Aerobrake will also enormously simplify the synthesis of multiple remote sensing spacecraft data sets. Combining observations from multiple instruments is a common research practice that can enormously increase the science available from past missions by leveraging synergies between data sets. Unfortunately, this practice is often time-consuming, difficult, and imprecise. But Aerobrake will be able to ameliorate these problems. For example, assume a researcher wants to combine the projected CCD data mentioned above with data from a single-row, “pushbroom”-style hyperspectral imager. Such imagers maintain a fixed pointing while the spacecraft moves; they scan (“pushbroom”) an area on the surface over a single exposure, and each element in the detector row may simultaneously observe in multiple wavelengths. The Aerobrake database would treat an observation from this pushbroom imager much like a CCD image. It would store observational data from each element of its detector row in separate entries. These entries would include multiple values associated with specific wavelengths and would be associated with relative position values, instrument definitions, and instrument and spacecraft state information. The researcher could then import the projection mappings from the projected CCD data and use them to perform the same projection of the hyperspectral imager data onto the same DEM—and, should they choose, the same interpolation as well.

Part 3: Technical Objectives

The proposed Aerobrake project requires two related but independent *domains* of technological development with domain specific challenges: the (1) *cloud database* and (2) *researcher Application Programmer Interface* (API).

Cloud Database

Cloud storage and processing are maturing but well explored technologies. However, as noted in the sub-topic solicitation, there is a substantial gap between the use of such technologies in commercial applications and their use for research with spacecraft data. We don’t propose to make significant advancements in the field of cloud computing, generally, but we do propose advance the state of the art for use of spacecraft remote sensing data of planets within a cloud computing environment. Our method for doing so relies on formatting the data within a traditional relational database in a novel way, treating each pixel (or footprint or equivalent smallest unit of sensing) as its own, independent observation. This maximizes the flexibility of the data and, from that point, all cloud computing and database techniques and

technologies can be brought to bear, providing researchers with new and powerful ways to utilize such data. While we have broadly defined the format and have a general sense of what new capabilities it will offer, a number of questions remain which will need to be addressed in Phase I.

Representing Data

We've broadly outlined how we expect most remote sensing data to be represented in the data; that is, as its smallest meaningful unit (a "pixel" or analogous) given context by a large number of relational mappings to metadata. We do not, however, want Aerobrake to be limited to only representing discrete or pixel-like data sets. While we are confident that if it can be represented digitally, then it can be represented within Aerobrake, we cannot solve the general problem of how to represent any possible kind of data within the Aerobrake schema. But we should be able to define standards for representing common types of data. In particular, we need to define a way in which to represent continuous data or data defined by by functions (e.g. geoids, spherical harmonics, maximum likelihood functions) which do not necessarily have an intuitive choice for "smallest meaningful units" to form the basis of data sets.

Data and Processing Volume and Performance

The initial representation of the raw data in what we are calling the Raw Data Layer (RDL) should be equivalent or very nearly equivalent in total size to the uncompressed representation of the same data in any other format. However it is our intention that each Derived Data Layer (DDL)—every calibration or map projection—likewise be stored permanently in the cloud. This could easily result in explosive increases in storage needs, and will certainly dramatically increase the complexity of the database. It is possible that the cloud database should only permanently store the most popular DDLs, while other DDLs might require special permissions to make and might only be retained temporarily. There is almost certainly going to be a trade-off between time, complexity, and real costs of storing data permanently vs. recreating it as needed, and that trade off needs to be explored. We might find, for instance, that only the most computationally intensive DDLs should be permanently stored, because the overhead associated with dynamically generating less computationally intensive DDLs is much less than the overhead associated with simply storing them.

Algorithms

As alluded to in the solicitation sub-topic text, adapting existing data processing and analysis tools for a cloud environment could represent a substantial effort. We aren't proposing to solve this problem, but an honest estimate of the level of effort required to adapt common types of data sets needs to be made in order to estimate the total resources that would be required to adapt the complete corpus of NASA planetary science data to Aerobrake.

Costs

Costs of cloud computing scale as a function of both total static data volume and processor hours. It is standard practice in cloud computing applications to benchmark performance and estimate cost based upon representative samples of data and processing, and optimize operations against cost when possible.

Researcher API

One of the reasons that researchers prefer to store voluminous amounts of raw data locally is because of the tremendous flexibility that they have in using the data with their own custom analysis tools. Among planetary science researchers, these tools have historically been written in the Interactive Data Language (IDL), though Matlab is also popular and Python / NumPy is quickly gaining traction. These languages are popular among planetary scientists because they are *array-based,* which makes them particularly efficient and well-suited for the manipulation of raster or image type data. The first step for almost all researchers, therefore, is to find a way to represent their data as some kind of an array. However, popular scientific data formats are generally optimized for sequential storage or other operational concerns, rather than for subsequent analysis. As such, they rarely take this need into account, and so a substantial amount of effort can go into simply putting the data into a form that is usable by tools written in the most popular languages. As such, we want to make cloud based data as quickly useful and available to researchers as possible by providing them with in-language modules or Application Programmer Interfaces (API) that interpret in-language commands as queries on the cloud databases, returning in-language objects over the internet that contain the requested information and are immediately usable by the researchers.

User Experience (UX)

We expect that the majority of users will interact with Aerobrake through a module or Application Programmer Interface (API) implemented in their research language of choice—most likely IDL, Python, or Matlab. That is, they will not directly access the database through a database front end, but they will formulate commands within their programming environment that the API interprets as database actions, executes on the server, and parses into responses formulated as in-language data objects. As such, the Aerobrake API must enable a wide range of potential user interactions with the Aerobrake database, including search, synthesis, retrieval, and potentially modification. It is important that the API serve as a straightforward abstraction of such operations; it needs to simplify these things and not make them harder. What the nature of such commands and interactions should be and how they should be implemented is an important open question. In keeping with generally-recognized contemporary software design best practices, we believe this question can be best answered by iterative prototyping and feedback from a set of representative end users. In addition to the usability of the API commands themselves (“the User Interface” or UI), the structure and format of any returned data objects is an important element of the overall UX which needs to be explored and addressed.

Permissions: Query vs. Modify

It would be, at the very least, unwise to allow anyone to Aerobrake modify the database at will. Even in the best case, this would lead to rapidly-increasing data volume and complexity, with a large fraction of the database consumed by “junk data” resulting from failed or nonsense investigations. We need to investigate and establish strategies for mitigating this problem while still allowing researchers to creatively draw on the power of cloud computing. One possibility is that individual researchers are given some usefully large amount of personal, private storage on

the cloud system (perhaps 10 Tb) in which to test and develop MapReduce [15] and other data processing algorithms. They could then apply or petition to some curatorial board to have those algorithms and the resulting DDLs propagated to the whole database and added to the permanent corpus. The identification or development of a viable method to allow individual researchers substantial flexibility in utilizing the cloud environment while maintaining the integrity of the platform is a critical component of establishing the feasibility of Aerobrake.

We expect a substantial difference in the complexity of implementation for API commands that simply query the database, as compared to commands that actually modify the database. While the former is likely to only require simple database search functions, the latter is likely to require support for frameworks such as MapReduce. Determination of the level of effort required to support all desired functions is an important step in assessing feasibility. This question is also closely tied to the *permissions* problem discussed above; that is, to determine if all desired functions are supportable, we need to define which functions are desired.

Bandwidth

It will be possible, even easy, for users to make requests which result in large volumes of data being returned over the internet. This has potential cost and usability implications on both the server and client sides, and so bandwidth costs needs to be estimated. In particular, modeling and prototyping of expected common query types needs to be done to determine if bandwidth costs (in terms of both price and availability) might make Aerobrake unworkable in a realistic research setting.

Part 4: Work Plan

The goal of the Phase I effort will be to assess Aerobrake's scientific, technical, and commercial merit. We will accomplish this primarily by creating a series of rapid prototypes, simulations, and models which will allow us to make performance and usability tests that address the outstanding technical questions noted above, and secondarily by drawing upon our experience with software development and project management. It is not our intention that the rapid prototypes will, together or independently, compose any kind of fully featured prototype to serve as a deliverable to NASA; as suggested in the sub-topic text, we expect to create a fully featured prototype demonstration during Phase II. But the source code, procedures, tests, and outcomes of work in Phase I will all be retained and recorded for review upon request. We will also develop and maintain a Concept of Operations (ConOps) document as the repository of information about software design, impact, feasibility, and other issues during Phase I. The final contract deliverable will be a report that will draw heavily on the ConOps and outline the work performed; the test results; the scientific, technical, and commercial merit and feasibility of Aerobrake; its relevance and significance to NASA needs; and strategies for developing the technology into a product for NASA or other customers.

Concept of Operations

Where possible, we will apply software systems engineering best practices to the design process. This will ensure that any Phase I products will be of maximum utility for Phase II efforts. As the reference point for best practices, we will use the IEEE book "The Project Manager's Guide to Software Engineering Best Practices." [10] We will generate a System Requirements

Specification (SRS) document that tracks the functional, performance, and interface requirements of Aerobrake as well as any design constraints (imposed by, for example, expected operating environments). We will then develop a Concept of Operations (ConOps) design document with IEEE Standard 1363 format. As defined by this Standard:

> *a ConOps is a user-oriented document that describes system characteristics for a proposed system from the users' viewpoint. The ConOps document is used to communicate overall quantitative and qualitative system characteristics to the user, buyer, developer, and other organizational elements (for example, training, facilities, staffing, and maintenance). It is used to describe the user organization(s), mission(s), and organizational objectives from an integrated systems point of view.* [11]

The ConOps is the central repository of information about the proposed system. It will primarily consist of text, describing user experience narratives, potential operating environments (a category which includes potential customers), operational constraints, and possible system limitations. We will also document considered but rejected design ideas along with the reasons for rejecting them. We may also include tables, diagrams, or mock-up images as required to fully communicate design concepts. Development of the ConOps will begin immediately upon the start of the project and involve all project participants, but it is a living document that will be continuously modified throughout Phase I and form the core of the Phase I final report.

For all Aerobrake development, we will use some kind of noSQL ("not only SQL"), most likely the popular and well supported Hadoop framework which utilizes the MapReduce query framework. Initial, small scale testing, especially in the first two months of Phase I, will occur in local (non-cloud) database instances; later and larger scale tests will make use of Amazon Web Services (AWS) cloud computing platforms, with particular reliance on the Elastic Compute Cloud (EC2), Simple Storage Service (S3), and DynamoDB.

Database Systems

The first concern for the Aerobrake database is the effectively representation remote sensing spacecraft data. We will use a local database instance as a testbed for developing techniques to efficiently and sensibly store a variety of common data. This will require the development of standard definitions for common categories of data. Such standard definitions are important both for API development and later projects to acquire and clean large corpuses of data. For instance, pixels and similar detectors arranged in arrays are very important and will be relatively easy to implement. They can be represented as we described above—elements with intensity, position values, and links to state- and instrument-related metadata. We will also begin to develop and maintain a manually populated dictionary of metadata categories (which might eventually be heuristically definable on arbitrary data sets), including tuple length ranges, data type information, and links to commonly related metadata categories. This database will not be limited to storing observational data as such. Relational data is also important, and we will begin to implement storage for some common categories of functions, maps, and models—projections and calibrations, for instance. Most of these will probably be easily storable as perfect hash functions or other optimized search structures, arrays representing systems of linear equations, or other common data structures.

We will download and use a set of test data from the PDS and represent them within a local database instance using the previously described paradigm of smallest meaningful units attached to metadata. In particular, we plan to use data from the Lunar Orbiter Laser Altimeter (LOLA) [12] and Diviner Lunar Radiometer Experiment (DLRE, or just “Diviner”) [13] instruments on the Lunar Reconnaissance Orbiter (LRO) [14]. Laser altimetry is an interesting use case because the “pixels” in question are not necessarily square or regularly spaced, but the functional form of the data is almost always interpolated and gridded. Diviner, likewise, is a good test data set because, while it is quite large, the raw data is stored in simple and well-documented ASCII tables, and the instrument itself contains a number of common but complexity-adding attributes—it is a hyperspectral, pushbroom-style with multiple boresights. Additionally, the Experimenter Data Record (EDR, i.e. raw) and and Reduced Data Record (RDR, i.e calibrated) are co-aligned and losslessly derived, which gives us the opportunity to represent relational mappings between raw and derived data without the need to use, develop, or port some version of the actual calibration software. Diviner and LOLA data are frequently used in concert, so we will also be able to explore a common case of synthesized data analysis. We will also explore multiple paths for representing continuous data within the database, with the specific case of a lunar geoid model.

We will use this local database to experiment with using relational data on observational data to conduct projection and calibration operations using MapReduce. MapReduce [15] is a common programming model for distributed, highly parallelized processing of large datasets that first gained prominence as the infrastructure for Google's World Wide Web indexing system. MapReduce techniques are predictably scalable using simulation packages like MRSG and MRSim; we will be able to estimate performance requirements as a function of database size and complexity without needing to load the full corpus of PDS data. This will enable us to make rough level-of-effort, performance, and cost estimates. As in any computing system, both storage space and processing power will influence Aerobrake's overall performance. Though we will not make specific attempts at database optimizations in Phase I, we will attempt to identify potential performance bottlenecks and "sweet spots."

When we are confident that we understand the performance (and therefore cost) requirements of our database through local and small-scale AWS benchmarking, we will migrate to AWS for larger scale and cloud-based testing with data volumes in the range of 1-10Tb (the approximate size of a single Diviner data delivery).

Application Programmer Interface

The Aerobrake API is the technology that will enable researchers to interact with data stored in the Aerobrake database. We will develop it concurrently with the Aerobrake database; choices about data storage standards will influence choices about API features and vice versa. Although we plan to eventually implement a multi-language API, we will prototype it in Python. Not all specific features of the Python prototype (such as object mutability and dynamic typing) will be adaptable to other languages, but broad principles of the software design will be.

Initial prototyping will involve developing command line-style interfaces for both simple and complex query categories such as simple cuts (for instance, the upper left quadrant of a CCD image) or complex cuts which rely on conditionals across multiple levels and kinds of derived data (for instance all of the map projected data within a given time range for which the

raw source data values are below some number). We will also prototype methods for interfacing with Aerobrake's MapReduce framework—defining calibration, projection, and cross-instrument data correlation commands, for instance.

Our initial prototype will allow us to test and/or model several aspects of Aerobrake's use costs and usability. Notably, we will be able to predict download data volumes for common sample queries and test some methods for managing data permissions. AWS has robust support for permission management via its web-based AWS Identity and Account Management (IAM) service. AWS IAM alone may prove to be sufficient for Aerobrake's users' needs and the curatorial needs of the Aerobrake database. Not only is it important that we are *able* to implement needed features, but that they are usable and useful for the conduct of actual scientific research. As such, our two scientific collaborators—Horgan and Rice—will test these prototypes and provide us with feedback from the perspective of actual target end-users.

Schedule and Management

Broadly, the schedule for Phase I will be as follows: in the first few weeks, we will establish overall project goals with the creation of the SRS and ConOps documents. We will then, over the first few months, perform small-scale, in-house prototyping of the database and API. This will phase into larger scale, cloud-based testing on AWS in the later months and culminate with the creation of the final deliverable—a report which describes Phase I work and assesses project feasibility and merit.

Month #	1	2	3	4	5	6
ConOps Creation						
Standalone Database Prototype						
Cloud Database Prototype						
Cloud Performance Testing						
Python UI Dev / Test						
UI / Database Testing						
Feasibility Report						

Figure: Phase I project timeline by task.

The three software developers on the project—Million, St. Clair, and Blandford—have largely overlapping skill sets, and each is qualified to perform most of the software development tasks on this project. The Project Management role will be filled by Million; Million and St. Clair will be primarily responsible for writing the final report and curating the ConOps document. With these exceptions, tasks will be split between the developers on an as-needed basis. Developers are expected to move between tasks, both to assist in meeting project goals and to maintain information flow throughout the team. Between the three developers, approximately 52 person-

hours per week are budgeted through Phase I. This is sufficient to assess the feasibility and merit of the Aerobrake system, as outlined above, and also allows for the exploration of currently unrecognized feasibility and merit issues as they arise.

	Estimated LoE (hours)
ConOps Creation and Curation	**100**
Standalone Database Prototype	**200**
Cloud Database Prototype	**250**
Cloud Performance Testing	**200**
Python UI Dev / Test	**250**
UI / Database Testing	**200**
Feasibility Report	**50**

Figure: The estimated Level of Effort (LoE) required for each broad task in Phase I in terms of developer hours.

The collaborators and contractors will work remotely from the PI at their respective home offices and institutions. The PI has experience successfully managing remote, small-team projects of this kind. In particular, the PI has successfully managed contracted projects involving Co-I St. Clair and Contractor Blandford with this arrangement in the past. The fact that the Phase I personnel are not co-located is not expected to be a problem. The PI will coordinate and supervise work with the collaborators and contractors by frequent contact through email, text (gchat/IRC), telephone, and video conferences. Coordination of software development and documentation will also be facilitated by collaboration and version control software, including Google Docs and Git.

Part 5: Related R/R&D

By the Offeror

The PI, Chase Million, conceived, developed, and is in an ongoing contract by the Mikulski Archive at Space Telescope (MAST) at Space Telescope Science Institute (STScI) to create gPhoton, a large database and associated toolkit for the archival of photon level data from the Galaxy Evolution Explorer (GALEX) mission. [16] GALEX was a NASA ultraviolet all-sky survey mission which ended in 2013 [17]. The ultraviolet detectors on the GALEX spacecraft were non-integrating micro-channel plates that recorded individual photon events with a time resolution of five thousandths of a second. [18] Due to constraints on budget, storage, and time, the GALEX data was only released by the mission team as integrated images except by special request, making it difficult if not impossible to mine the GALEX data for short time domain variability. Mr. Million designed and is in the final stages of developing a standalone calibration pipeline and ~100 Tb database for creating and storing GALEX photon-level data. To facilitate use of this database, the PI has also created a set of Python based tools

which query the database and generate calibrated light curves, images, and movies directly from the photon-level data on user request. While gPhoton is a wholly distinct, mission-specific project that does not draw on "cloud" technology, it shares many philosophical fundamentals with Aerobrake. It attempts to maximize the flexibility and total utility of a data set by describing it in terms of small flexible, units. It offloads large storage and processing tasks to an integrated database and provides end-user tools for utilizing that database. The PI will draw upon his experience in the development of gPhoton to inform the development of Aerobrake.

By Others

To our knowledge, Aerobrake is not competing directly with efforts by any other groups. However, others have certainly made attempts to tackle the same kinds of problems or draw on the power of similar technologies to enhance the scientific return from NASA and planetary science missions.

Data Search and Retrieval

The need to locate data of interest in archives is as old as the existence of archives. Within planetary science, notable graphical data browsers include JMARS [4] Google Mars, PILOT [6], and PDS ODE [6]. Many instrument and mission teams have their own toolkits for locating and browsing data of interest. For example, the Maestro rover planning software [**cite**] is often used in this way by science team members. Such toolkits are sometimes made available to the community as a whole. All such tools can be thought of as, in essence, *metadata* browsers. As such, PDS's recent upgrade to version 4, which defines an eXtensible Markup Language (XML) metadata format, should dramatically improve the usability of such data [3].

Data Processing

The most commonly used software tool in planetary cartography is the Integrated Software for Imagers and Spectrometers (ISIS), which was developed by the USGS and is now in its third incarnation, ISIS3 [7]. For supported instruments, ISIS can generate a number of standard research-grade derived data products including calibrated, orth-rectified, and mosaicked images. Many mission teams also heavily rely on the Video Image Communication and Retrieval (VICAR) software suite maintained by the Multi-mission Instrument Processing Laboratory (MIPL) at the Jet Propulsion Laboratory (JPL), although it is perhaps less well-known in the research community as a whole. It has been developed to process imaging data, primarily from JPL-run missions. [19]

Just as with search tools, many mission and instrument teams internally develop unique data processing software, which is sometimes eventually released to the community. An exhaustive list would be quite long, but notable examples include the CRISM Analysis Toolkit (CAT) [8] and the OMEGA Reduction Software (a bare-bones calibration pipeline released by the OMEGA team along with the raw data) [9] As noted previously, many mission and instrument teams choose to simply release calibrated, reduced, or projected data products without the accompanying code or documentation necessary to recreate the raw data from them.

Cloud Computing

The state of cloud computing in planetary science is explained in the Summary document for the 2012 Planetary Data Workshop [2]:

> Cloud computing and storage of planetary data are being investigated by NASA engineers, and current low-risk options include its use as an operational, secondary copy of digital data and a data access point. For example, the NASA Mars Exploration Rovers mission is effectively managing their data sets through use of cloud computing and data storage mechanisms.

In other words, cloud computing is still considered an investigational technology in planetary science and is only used in limited capacities. Notable examples include the USGS Projection On the Web (POW) project, which allows users to create and download map projected data products for ISIS supported instruments through a web inteface [20], and a separate project that runs ISIS3 in the Amazon Cloud. [21]

Database Technology

Object Oriented Data Technology (OODT) is an open-source data management system framework developed by JPL and now managed by the Apache Software Foundation [22]. It is widely used in both planetary science and other disciplines for the discovery and query of distributed data resources. OODT is a component-based software system with elements that may prove useful in the development of Aerobrake.

Part 6: Key Personnel and Bibliography of Directly Related Work

Chase Million (Principal Investigator) is the CEO and owner of Million Concepts LLC, a business which has sought to improve the quality and availability of software engineering in research science since 2011. He has a Bachelor of Arts in Physics from Cornell University (2007) and ten years of experience as a researcher and software developer working with spacecraft data in planetary science and astronomy. In this capacity, he frequently fills a project management role that includes the responsibility of ensuring that work proceeds according to contract agreements. Prior to starting Million Concepts, Mr. Million was the lead software engineer for the ground calibration pipeline software for the GALEX mission at the California Institute of Technology. Prior to that, he was a member of the Mars Exploration Rover Pancam team at Cornell University. He also served as the team lead for the design and construction of an automated ground station for command and control of a CubeSat class satellite developed by Cornell University. He has direct experience conducting or supporting research with a number of datasets from NASA planetary missions which include the Mars Exploration Rovers (MER), Mars Reconnaissance Orbiter (MRO), Odyssey (ODY), and Mars Global Surveyor (MGS), as well as general expertise in spacecraft engineering, data management and calibration, and research support.

Mr. Million has experience serving as the primary point of contact with contracting organizations and success leading small teams to ensure that projects proceed according to contract terms. As the PI, he will serve as project manager and be involved in or oversee all aspects of Phase I work. He will work from his home office in State College, PA. His total time commitment over Phase I will be approximately 40% (0.4 FTE) over six months, or approximately 64 hours per month for 384 hours or 2.4 months over Phase I. As owner and CEO of Million Concepts LLC, Mr. Million is primarily employed by the SBC and therefore

eligible to be the PI of this SBIR proposal. Million Concepts LLC has an ongoing contract with Space Telescope Science Institute—to develop a large database and associated tools to archive and provide access to the GALEX photon level data and described more thoroughly in Part 5—for 0.4 FTE (or 4.8 months) of his time through the end of 2014. He has three additional pending grants to NASA and NSF that, if funded simultaneously, would commit 5.25 months of his time over the following year, though it would be unusual for all pending proposals to be funded given historical acceptance rates in those programs, and funding decisions for several are not expected until mid- to late-2014.

Dr. Michael St. Clair (Co-Investigator) is the Chief Technology Officer of Million Concepts LLC. He has a Bachelor of Science in Mathematics and a Bachelor of Arts in Theater from Case Western Reserve University (2005), as well as a Ph.D. from Stanford in Theater and Performance Studies (2013) for his research into the social and technological aspects of play and games. Dr. St. Clair is a technologist who has performed research and programming tasks for a wide variety of design, engineering, and scientific applications ranging from quantum computing to musical instrumentation. He will work from his home office in Mountain View, CA but be employed by Million Concepts LLC as one of the software developers for Phase I and will also assist with project definition and documentation tasks. His total time commitment over Phase I will be approximately 40% (0.4 FTE) over six months, or approximately 64 hours per month for 384 hours or 2.4 months over Phase I.

Part 7: Relationship with Phase II or Future R/R&D

As suggested in the solicitation text for this sub-topic, we will develop a prototype demonstration in Phase II. More specifically, in Phase II we expect to integrate multiple mission data sets into an Aerobrake cloud database system, conduct a beta test of the API with users from the planetary science community, and release public documentation and source code for the Aerobrake system to encourage other researchers and mission teams to port their repositories. Through conferences and personal contacts, we will also actively pursue collaborations with potential NASA stakeholders, including past and current mission teams, archival groups (PDS, USGS Astrogeology, and the Regional Planetary Imaging Facility Network), and individual researchers. We believe that the feasibility study that will arise from Phase I is an important product to have on hand before soliciting such stakeholder buy-in. By the end of Phase II, we will have resources in place that will be useful—and, we hope, being used—for planetary science research.

While Aerobrake is ultimately a commercial venture of the type that SBIR is intended to support, we also strongly believe that software used for scientific research should be open-source and free whenever possible. During Phase II, we will develop “in the open,” on a public version control repository such as Github, and make all of our code available to the community on a continuing basis. (Note that the Aerobrake source is likely to incorporate other Open Source software, the licenses for which may *require* that we share in kind.)

We are utilizing Amazon Web Services (AWS) in Phase I as a platform for prototyping Aerobrake because it is a readily available, widely used, relatively cheap, and industry standard ecosystem, and we expect to continue to develop on Amazon in Phase II for the same reasons. However, there is nothing about the Aerobrake concept which confines it to the Amazon cloud,

and we will develop it with the intention that the system could eventually utilize or be migrated to NASA-owned storage and computing assets.

Part 8: Facilities/Equipment

Million Concepts LLC is operated from an office in State College, PA. Several computer systems are available with multiple operating systems and high-bandwidth internet. In order to develop Aerobrake within a representative cloud computing environment, we will purchase cloud computing services from Amazon Web Services.

Part 9: Subcontracts and Consultants

Mike Blandford (contractor) has a Masters in Computer Science from Indiana University, Bloomington (2007) and 7 years of experience in commercial software development. He has worked on a wide range of software development projects including databases, server side algorithms, and User Interface (UI) and API development. Mr. Blandford will serve as a software developer during Phase I and provide consultation and expertise related to databases and APIs. He and will commit 10 hours per week or 240 hours over the six month project.

Dr. Melissa Rice (consultant) is a Planetary Geologist whose research focuses on the surface composition and landscape evolution of Mars. She obtained her Ph.D. in Astronomy from Cornell University (2012), and is currently a NASA Astrobiology Institute (NAI) Postdoctoral Fellow at the California Institute of Technology. She is a Science Collaborator on the Mars Exploration Rover (MER) and Mars Science Laboratory (MSL) missions and has extensive experience with the acquisition, processing, calibration, synthesis and analysis of spacecraft data. Specifically, she uses imaging, topography, and multi/hyperspectral datasets from MER Pancam, MSL Mastcam and ChemCam, Mars Global Surveyor (MGS) MOC and MOLA, Mars Odyssey (ODY) THEMIS and Mars Reconnaissance Orbiter (MRO) CTX, HiRISE and CRISM. For her research, Dr. Rice performs radiometric calibrations of spectroscopic data, spatial projections of imaging data, creation of digital elevation models (DEMs) from stereo image pairs, and fusion of hyperspectral data with high-resolution imagery. Dr. Rice will provide feedback and input on rapid prototypes during Phase I to ensure that Aerobrake appropriately addresses the needs of working research scientists. Her term of appointment at the California Institute of Technology ends on 4/31/2014, after which she will be an independent consultant. Her Phase I commitment will not exceed 20 hours.

Dr. Briony Horgan (collaborator) is an Assistant Professor of Earth, Atmospheric, and Planetary Sciences at Purdue University. She obtained her PhD in Astronomy and Space Sciences from Cornell University in 2010 and served as an Exploration Postdoctoral Fellow in the Mars Space Flight Facility at Arizona State University until last year. Her expertise is near- and mid-infrared spectroscopy of planetary surfaces from orbital and landed platforms, and she has lead the creation of local, regional, and global-scale hyperspectral image mosaics of both Mars and the Moon, using data from the Mars Express OMEGA, Mars Reconnaissance Orbiter Compact Reconnaissance Imaging Spectrometer for Mars (MRO/CRISM), and Chandrayaan-1 Moon Mineralogy Mapper (M^3) imaging spectrometers. Because of their 3D nature, large size, and the high level of processing required for calibration and analysis, hyperspectral image cubes

present unique challenges for data processing, management, and storage that could be successfully addressed by Aerobrake. Dr. Horgan will provide input and feedback on design documents and rapid prototypes during Phase I to ensure that Aerobrake appropriately addresses the needs of working research scientists. Limited participation in collaborative and exploratory activities like this project are expected as part of her academic year commitment, so no funding is requested for her Phase I commitment of not more than 20 hours.

Part 10: Potential Post Applications

Our revenue model for Aerobrake will follow that of open-source software companies like Redhat and Rasdaman. These companies which release their tools for free but charge for support, targeted development, and some commercial uses. We further believe that having Aerobrake in the portfolio of Million Concepts LLC will increase our visibility and attractiveness to both academic, government, and commercial spacecraft data users as an outside vendor and contractor for custom research and analysis software; these services are at the core of our business model. We anticipate future inclusion of Aerobrake in mission data management plans from very early in the design process; we also expect users to port many existing data sets. Long-term, Aerobrake could form the basis for a new NASA planetary data archive that fills an existing gap in services and support.

While Aerobrake is targeted at orbital spacecraft data of planets other than Earth, it is worth noting that the techniques are wholly adaptable to terrestrial orbital data sets. NASA has a robust and vibrant earth-sensing program which could potentially make use of Aerobrake, and the software may also be of interest to the military for geospatial awareness applications. The uses of such terrestrial orbital data sets are myriad—they include agriculture, resource utilization, climate monitoring, and military intelligence. There has been intense recent private-sector interest in orbital remote sensing from companies including Google, Planetary Resources, and Skybox Imaging, all of which are potential Aerobrake customers.

Furthermore, many of the techniques and concepts behind Aerobrake are extensible to non-orbital remote sensing data generally. That is, the only thing special about orbital data is that the instrument is pointing, roughly, “down.” The same data format and retrieval paradigms could apply equally as well to data taken from instruments “on the ground” so long as the camera definition and state are well understood or can be derived. With recent advances in the reconstruction of camera position and orientation from multiple images of the same scene by, for example, Microsoft, it’s possible that this includes a large fraction of *all* imaging data. We don’t know what additional capabilities an Aerobrake-like system would add to the exploration of such data, and such uses are many years out, but multidisciplinary possibilities ranging from instant 3D modeling of art installations derived from photographs in museum catalogs to maps of defoliation patterns derived from large public corpuses of geotagged photographs cannot be discounted.

Part 11a: Essentially Equivalent and Duplicate Proposals and Awards

None.

Part 11b: Related Research and Development Proposals and Awards

gPhoton: a project to archive and make available the GALEX photon level data

The PI, Chase Million, developed and is in an ongoing contract by the Mikulski Archive at Space Telescope (MAST) at Space Telescope Science Institute (STScI) to create gPhoton, a large database and associated toolkit for the archival of photon level data from the Galaxy Evolution Explorer (GALEX) mission. The ultraviolet detectors on the GALEX spacecraft were non-integrating micro-channel plates that record individual photon events with a time resolution of five thousandths of a second. Due to constraints on budget, storage, and time, the GALEX data was only released by the mission team as integrated images except by special request, making it difficult if not impossible to mine the GALEX data for short time domain variability. Mr. Million designed and is in the final stages of developing a standalone calibration pipeline and ~100 Tb database for creating and storing GALEX photon-level data. To facilitate use of this database, the PI has also created a set of Python based tools which query the database and generate user-requested, calibrated light curves, images, and movies directly from the photon-level data.

Despite the similarities, the gPhoton project should not be considered equivalent work either in whole or in part. The gPhoton project is heavily mission specific development of calibration tools for the GALEX photon level data in particular and is not directly extensible to any other data set. The gPhoton project also does not explicitly draw upon cloud computing technologies, only a traditional, static database, and while gPhoton includes the creation of python tools for the access and processing data, these could in no sense be thought of as an API.

Works Cited

[1] Hughes, J. Steven, and Susan K. McMahon. The Planetary Data System-A Case Study in the Development and Management of Meta-Data for a Scientific Digital Library. Springer Berlin Heidelberg, 1998.

[2] Gaddis, L. R., et al. “Summary and Abstracts of the Planetary Data Workshop, June 2012.” Working version online at astrogeology.usgs.gov/search/details/Docs/PlanetaryDataWorkshop/PlanetaryDataWorkshop_AbstractVol_Sept2013_Draft/pdf

[3] Neakrase, Lynn DV, et al. "The PDS4 archive: new structure, new possibilities." Third International Planetary Dunes Workshop: Remote Sensing and Image Analysis of Planetary Dunes. Flagstaff, Arizona, abst. Vol. 7049. 2012.

[4] Gorelick, N. S., et al. "JMARS: A multimission data fusion application." Lunar and Planetary Institute Science Conference Abstracts. Vol. 34. 2003.

[5] Wang, J., et al. "Planetary data access through the orbital data explorer from the PDS geosciences node." Lunar and Planetary Institute Science Conference Abstracts. Vol. 40. 2009.

[6] Bailen, M. S. “The PDS Planetary Image LOcator Tool (PILOT).” Summary and Abstracts of the Planetary Data Workshop, 2012.

[7] Torson, J. M., and K. J. Becker. "ISIS-A software architecture for processing planetary images." Lunar and Planetary Institute Science Conference Abstracts 28 (1997).

[8] Murchie, S. et al. Compact Reconnaissance Imaging Spectrometer for Mars investigation and data set from the Mars Reconnaissance Orbiter's primary science phase. Journal of Geophysical Research, 114, E00D07 (2009).

[9] Bellucci, G., et al. “OMEGA/Mars Express: Visual channel performances and data reduction techniques.”
Planetary and Space Science 54.7 (July 2006).
[10] Christensen, Mark J., and Richard H. Thayer. The project manager's guide to software engineering's best practices. IEEE Computer Society, 2001.
[11] IEEE Standards Association. IEEE 1362-1998 (R2007). “IEEE Guide for Information Technology—System Definition—Concept of Operations (ConOps) Document”.
[12] Riris, Haris, et al. "The lunar orbiter laser altimeter (LOLA) on NASA's lunar reconnaissance orbiter (LRO) mission." Defense and Security Symposium. International Society for Optics and Photonics, 2007.
[13] Paige, D. A., et al. "The lunar reconnaissance orbiter diviner lunar radiometer experiment." Space Science Reviews 150.1-4 (2010): 125-160.
[14] Chin, Gordon, et al. "Lunar Reconnaissance Orbiter Overview: The Instrument Suite and Mission." Space Science Reviews 129.4 (2007): 391-419.
[15] Dean, Jeffrey, and Sanjay Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters.” OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.
[16] Fleming, Scott W., et al. "New Kepler Data Products At MAST For Stellar Astrophysics." American Astronomical Society Meeting Abstracts. Vol. 223. 2014.
[17] Martin, D. C., et al. “The Galaxy Evolution Explorer: A Space Ultraviolet Survey Mission.” Astrophysical Journal Letters 619.1 (2005).
[18] Siegmund, Oswald, et al. “High performance microchannel plate imaging photon counters for spaceborne sensing.” Proc. SPIE 6220, Spaceborne Sensors III, 622004 (May 30, 2006).
[19] Chien, Steve A. "Automated synthesis of image processing procedures for a large-scale image database." Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference. Vol. 3. IEEE, 1994.
[20] Hare, Trent. “Map Projection on the Web (POW): ‘ISIS3 in the Cloud.’’ LPSC 44 (March 2013).
[21] Laura, J. (2012). "ISIS3 in the Amazon Cloud." Poster Presentation at the 2nd Annual USGS Planetary Data Users Workshop. June 25-29, 2012.
[22] Mattmann, Chris A., et al. "Software architecture for large-scale, distributed, data-intensive systems." Software Architecture, 2004. (WICSA 2004). IEEE, 2004.