1. Background
Some big data applications are characterized by workloads that generate data “hotspots”, where the frequency of data access varies based on data values/regions. For example, for geospatial datasets, certain spatial regions may be of more interest than others. A geophysical dataset covering, say, the San Andreas Fault may witness hotspots because certain parts of the San Andreas Fault are studied more intensively than others. Different regions in the data may become hot sets at different points in time, based on a variety of external factors, such as the occurrence of certain events, e.g. an earthquake. In big data applications, is it feasible to provision the dataset, i.e. assign storage and computational resources to data, such that there is differential provisioning between the overall (cooler) dataset versus for the hot spots in that dataset. Detection of such “hot spots” (for spatial data) or “hot periods” (for temporal patterns) could be interesting since that may imply that the overall dataset could be provisioned with lower performing hardware while the “hot spot” could be provisioned by higher performing (more expensive) hardware. Once again, these hot spots are regional and time-varying.

While our example is based on a particular application that deals with remote-sensing data in the geosciences, we believe the notion is applicable to a broad class of spatiotemporal data.
2. Example access patterns and workloads

The data in our example are related to earth science, specifically airborne LiDAR data (Light Detection and Ranging) data from the NSF OpenTopography data facility. Based on an analysis of data access logs, we can detect both spatial and temporal “hot spots”, which are caused either by many users accessing the same data, or a few users repeatedly analyzing data in a given spatial region or in a short time period.

The maps in Figure 1 show the spatial data access patterns for two data sets—one related to the San Andreas Fault (left) and the other for Lake Tahoe (right). The maps show the boundary of the datasets (orange region on left; black outline on right) and the frequency of data access—the red regions are more frequently accessed.

Rather than using caching schemes that are oblivious to application semantics, there is an opportunity here to provide application-based hints to assist with how the system apportions resources to part of a big data dataset. The datasets here are in the 10’s to 100’s of TB, with hot sets being 100GBs or a few TB’s. Regions that are more heavily accessed and processed could receive more resources than those that are not as heavily used. How do such workload patterns differ among data genres and application types? While there may be some big data applications where all data need to be accessed with more or less equal frequency, we believe there are also applications where there is an opportunity for some amount of differential implementation based on the hot spot phenomenon.

Big data systems may require multiple storage classes within a given system, for example, hard disk, SSD, and RAM filesystem, with implicit as well as explicit schemes for moving data among these classes. The big data system should be capable of exploiting these multiple levels of storage efficiently and transparent to the user. Some of this could be achieved via simple caching schemes, while in other cases there may be an opportunity for a more thorough analysis of data access logs and for schemes that are more strategic. For example, there may be cases where a large dataset is stored in Hadoop (to, say, enable rapid upload of data and execution of basic subselection and filtering operations), while a subset of that data, which is the hot spot, may be implemented in, say, a database system (to exploit complex SQL and query optimization). Note that traditional hierarchical storage using nearline storage (e.g. tape-based systems) will not work any longer, since users are indeed interested in running “big data” applications, i.e. applications that may sweep through large swaths of the entire data collections, while they also run applications that create hot spots.

What is the cost of “spinning up” smaller subsets of data in on-demand databases (similar to the myDB concept mentioned some time back by Jim Gray), versus leaving the data in HDFS and utilizing the parallelism of MapReduce to meet the user’s performance expectations. How should one go about defining a workload for such scenarios and which aspects of a hybrid system should the workload test?