mrBox: Making MapReduce computations transparent for scientists.

MapReduce is a popular programming paradigm used to address massively parallel data-intensive algorithms. The main reasons behind MapReduce's success are (a) ease of use: MapReduce provides an intuitive way to express a task in a parallelisable form and (b) cost-effective: MapReduce clusters typically comprise off-the-shelf components/computers. While initially pioneered and used by search engines and Internet-related companies, MapReduce is currently being used in a wide variety of industries, including the financial and the scientific sectors. Aiming to support the long-tail of users and the citizen scientist, mrBox brings MapReduce closer to the user in a user-friendly and transparent way.

mrBox builds on and extends the user-paradigm of DropBox and of similar cloud-based services. Users are able to store files in a local folder, while this local folder is being monitored and synchronised with a remote HDFS store. Apart from an initial registration and logging-in, the user can therefore interact with a HDFS store as if it was local. In addition to this functionality, users can also submit hadoop jobs through special folders. Such special folders will start a Hadoop job as soon as they receive all the necessary computational components - i.e. input data as well as one mapper, one reducer and, optionally, one combiner script, and synchronises back, or links, the resulting data set. The initial version of mrBox accepts scripts in Python and makes use of Hadoop's streaming functionality. Points for further development and investigation are the following:

* Two-level re-direction and data tracking: mrBox could potentially be bound to multiple MapReduce (Hadoop/HDFS or other) backends. Additionally, it could also link to pure storage backends. This would require the design and implementation of an intermediate gateway that would facilitate a second level of mapping between local and remote resources. This gateway should also "reason" about best strategies in moving data between pure storage points and HDFS in a transparent way.
* Support linking to arbitrary remote file-systems. This could be facilitated, for instance, through integration with technologies such as SAMBA. Such functionality could be further extended by researching and implementing ways for 3rd-party transfers, while keeping appropriately placed links on the local folder, providing transparent access to users.
* Investigating mrBox as a server-side service. This action would explore server-side requirements and provide APIs to programmatically communicate with a server-side mrBox instance (e.g. potential use in OMERO).
* Improving data-movement efficiency for the Hadoop-based backend. By default HDFS accepts files over scp, which can be sub-optimal for large transfers. Furthermore, each file is checksummed before it is made available on the distributed store. This action would investigate the Hadoop internals as well as options for replacing the default transfer mechanism, without compromising the robust, parallel design of HDFS. Further, delayed checksumming mechanisms could be investigated as well as other optimisations to support relevant eScience use-cases.