[redacted]

3 MINUTE READ

Like many others, I continue to be impressed by the open-source Apache Hadoop software framework. With aplomb, it manages Big Data, images, documents, web logs–and tons of other unstructured and semi-structured data–for large, petabyte-scale processing on tasks ranging from search to machine learning to scoring and data refinement. Given Hadoop's considerable business potential, it's worth asking a few key questions: Where does Hadoop excel and fall short? How does it compare to existing technologies? And, what does Hadoop need, as an open-source project, to reach full maturity without fragmenting the code?

To see the right answers, you need both a starry and gimlet eye. Hadoop is a file system, great at handling unstructured data with economical scalability across few or many servers with a "capture in case it's needed" load mantra. Hadoop has a role in enterprise data architectures, but on its own it is not a database or an application, and it can be as inefficient with structured data as it is welcoming of unstructured data. Accessibility, security, management infrastructure, and disaster recovery are improvements on the open-source community developers' to-do list as they work passionately to bring Hadoop to maturity.

So how should we use Hadoop today? The simple answer is that there is no one-stop technology for all data and all analytics. I think you really need to take a "best of breed" approach that sets up the Hadoop file system as one of several in an architecture that works together on all data with different kinds of structure, or schema, and at various stages in the data pipeline. Developers and administrators have to think about a range of variables in balancing Hadoop alongside a relational database management system.

For instance: Hadoop is well suited for data with evolving or minimal structure. A relational database management system, meanwhile, is better at stable schema and structured data. Hadoop can handle raw, Big Data for batch analysis in a low-cost storage model. But a relational database can do iterative analysis of cleansed data efficiently stored. As you might guess, it takes a certain amount of nuance and artistry among the "data scientists," developers, and database administrators to get this technology mix right. That's one reason why my Teradata colleague Bill Franks has written that we need to change the title of "data scientist" to "data artist."

Hadoop is an open-source innovation with true business value. That means artistry and a great deal of care should go into responsibly moving the open-source project forward, balancing commercial interests of vendors with a commitment to the open-source community. Open source cannot mean open ledger into stack integration, custom software, and other value adds built on top of the platforms a company uses.

There is still a lot of work to be done. For instance, the lack of community consensus around a defined standard interface raises the unwelcome specter of UNIX-style fragmentation. For Hadoop to reach full maturity without fragmenting the code, it must be the joint effort of the collective data analyst and scientist community. At my company, Teradata, we make it policy to embrace open-source collaboration wherever and however we can. We converted to Linux some time ago and we contribute to open-source projects like Eclipse, PostGreSQL, Pentaho and, of course, Apache and HCatalog, as do our peers like Hortonworks (which spun out of Yahoo, where the Apache Hadoop open-source project began).

As the Hadoop open-source project matures, those of us who rely on it must find ways to give it the support it deserves. Every technology business should partner with the open-source developer community in their own way and support projects like Hadoop however possible. I think if we maintain a sensible corporate engagement in the open-source community, we can create conditions where we all do our best work.

[redacted]