HBase auf DXRAM

Einstiegspunkte für DXRAM in Hadoop & HBase

in Bearbeitung!

Department of Computer Science
Heinrich-Heine-University Düsseldorf, Germany

21. November 2018

Inhalt

- Motivation
- Hadoop Ökosystem und Hbase RegionServer
- So machen es andere
- Replacement, Integration oder Improvisation
- DXRAM hat eigene App
- DXNET für FS CRUD
- Stolpersteine
- Fazit

Motivation

- DXRAM: ein konkreter Anwendungsfall
- noSQL: schnelle Änderungen im RAM, HDFS nicht dafür ausgelegt
- HBase: HDFS als Persistenzschicht

Hadoop Ökosystem

- Prozessverwaltung
- HDFS, Scheme und Connector Konzept
- viele, speziell auf Blöcke in HDFS ausgelegte Methoden

Hbase RegionServer

- RegionServer für Key-Region zuständig
- RegionServer sind Apps, verwaltet durch RessourceManager von Hadoop/YARN
- RessourceManager verwaltet Nodemanager auf Hosts
- Apps (App/Master) laufen auf Nodemangern
- Host eines Dateiblocks bestimmt Wahl des zuständigen RegionServers

So machen es andere

Nur als Speicherpunkt und unverteilte Verarbeitung:

- ftp connector: Namenode hat auch direkt die Daten
- Google Cloud Storage Connector

Im Arbeitsspeicher:

- Alluxio (Hadoop Kopie; Mounten eines Connectors in Ordner)
- Ignite (App-Master und Nodemanager Replacement)

Replacement, Integration oder Improvisation

1. HBase Thrift Interface: Wenn Client nur das nutzt, kann man diese Objekte, Methoden, . . . auch ohne HBase und Hadoop implementieren.
2. Connetor: DXRAM ist kein verteiltes Dateisystem. Dies müsste man erst noch bauen.
3. Improvisieren: Hadoop lokaler Dateizugriff nutzen und DXRAM dort mountfähig machen

Replacement, Integration oder Improvisation

zu 1: HBase nachbauen?
zu 2: HDFS nachbauen
zu 3: . . . performance von libfuse :-(

DXRAM hat eigene App

Mir war nicht klar, wie ich mit DXNET direkt auf einen Peer von DXRAM zugreifen kann. Auch das Kombinieren des DXRAM Codes in einen Hadoop FileSystem Connectors war mir schleierhaft. Es gab nur DXRAM Anwendungen als Beispiel, welche als Anwendung auf einem Peer liefen.

DXNET für FS CRUD

- CRUD: Create, Read, Update, Delete
- HBase: append!!!
- DXNET als Austausch zwischen Hadoop FS Connector und DXRAM App
- hinterher wäre DXNET transfer nur auf localhost!?
- Falls RegionServer alleine den FS zugriff in HBase macht: Warum diesen nicht anpassen anstatt Connector zu bauen?

Stolpersteine: Bessere Vorgehensweisen

- FTP Connector Umbau auf Multinode, dann DXRAM Rest API ähnlich webdav
- Messung Multinode FTP: wo führt Hadoop Code aus
- HBase Replacement (Thrift API) in 3 Stufen: erst DXRAM Datenbank, dann Snapshot, dann Kompression
- Ignite: YARN gar nicht nutzen, RegionServer zu DXRAM App umschreiben?
- DXNET und DXRAM: IDL für Objekte, die sich nur aus elementaren Datentypen und Arrays (max. Länge) zusammensetzen

Nichts ist nerviger, als 3 Wrapper für 3 Systeme in beide Richtungen zu machen für jedes Objekt.

Stolpersteine: Prozess auf Block?

Ein `getBlock()` oder `setBlock()` ist in HDFS nicht zu finden! Arbeitet Hadoop primär mit mehreren Dateien, wo es mit `append()` im letzten Dateiblock arbeiten und macht dann ein kompliziertes `concat()` ?

Fazit

Hadoops Prozess- bzw. Ressourcen-Management ist zu stark an HDFS und dessen Blockverteilung gekoppelt! Ignite und Alluxio konstruierten daher auch ein Replacement! Außerdem: Ist es nicht leichter HBase mit DXRAM nachzubauen, anstatt DXRAM zu einem verteilten Dateisystem zu machen? Vermutlich Ja. **-> Apache Thrift**

Fazit

Aber: Alle von mir gefundenen Projekte werben mit einer EINBINDUNG in Hadoop, nicht aber mit einem ERSATZ. Ein art **DXRAM.Base** wäre aber sogar ein HBase Ersatz! Konkrete Anwendungsfälle, wo auf Hadoop bei Verwendung von HBase verzichtet werden kann, sollten in Zukunft gesucht werden.