Universidad de Granada / Instituto de Astrofísica de Andalucía

Integrating NoSQL Technologies into Virtual Observatory

Máster en Métodos y Técnicas Avanzadas en Física
Trabajo Fin de Máster presentado por José Antonio Magro Cortés

2013

Contents

Chapter 1

Overview

The volume of data produced at some science centers presents a considerable processing challenge.

At CERN, for instance, almost 600 million times per second, particles collide within the Large Hadron Collider (LHC). Each collision generates particles that often decay in complex ways into even more particles. Electronic circuits record the passage of each particle through a detector as a series of electronic signals, and send the data to the CERN Data Centre for digital reconstruction. The digitized summary is recorded as a "collision event". Physicists must sift through the 15 petabytes or so of data produced annually to determine if the collisions have thrown up any interesting physics.

CERN does not have the computing or financial resources to crunch all of the data on site, so in 2002 it turned to grid computing to share the burden with computer centers around the world. The Worldwide LHC Computing Grid is a distributed computing infrastructure arranged in tiers which gives a community of over 8000 physicists near real-time access to LHC data..

So we could think that current deployed technologies are maybe obsolete and a new rethink should be made. If CERN (but not just CERN as we will discuss later) has changed its storage policy (leaving behind the classic client-server model), why not change the way the data are accessed?

Almost a decade after E. Codd published his famous relational model paper, the relational database management systems (RDBMS) have been the must-be tools. Today, non-relational, cloud, or the so-called NoSQL databases are gaining mindshare as an alternative model for database management. NoSQL movement is growing rapidly. Often more characteristics apply such as schema-free, easy replication support, simple API, eventually consistent / BASE (not ACID), a huge amount of data (big data) and more.

In this document, we present an alternative, using one of the NoSQL databases free available, to offer a different way of challenging these problems, and always operating inside VO tools.

Chapter 2

Astronomy Big Data

Big data are high-volume, high-velocity, and/or high-variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization. Examples of big data include information such as Web search results, electronic messages (e.g., SMS, email and instant messages), social media postings, pictures, videos and even system log data. However, it can also include credit/debit card transactions, check images, receipts and other transactional information depending on the source of the information.

Astronomical datasets are growing at an exponential rate: high performance computing applications in astronomy are enabling complex simulations with many billions of particles, while the forthcoming generation of telescopes will collect data at rates in excess of terabytes per day. This data deluge, both now and into the future, presents some critical challenges for the way astronomers derive new knowledge from their data.

In this chapter, we make an overview of some of the greatest telescopes, the way they acquire and store data and the problem they face.

Figure 2.1: ALMA dishes in Atacama Desert

2.1 Atacama Large Millimiter Array

2.1.1 ALMA Instrument

ALMA is a worldwide project; the synthesis of early visions of astronomers in its three partner communities: Europe, North America and Japan.

ALMA is a fusion of ideas, with its roots in three astronomical projects: the Millimeter Array (MMA) of the United States, the Large Southern Array (LSA) of Europe, and the Large Millimeter Array (LMA) of Japan.

Millimeter Array

The origins of the Millimeter Array (MMA) are found in the science of the NRAO 36-Foot Telescope (later known as the 12-Meter Telescope), soon followed by the 4.9m telescopes at the University of Texas and Aerospace Corporation, the 14m telescope at the Five Colleges Radio Astronomical Observatory, and the 7m telescope at AT&T Bell Labs. The millimeter interferometers of the University of California (Berkeley) at the Hat Creek Observatory (later the Berkeley-Maryland- Illinois Association, or BIMA) and the California Institute of Technology at the Owens Valley Radio Observatory demonstrated the power that comes with high angular resolution for studying the sources found with the single dishes. The experience of using a powerful, flexible array that was provided by NRAOs Very Large Array (VLA) at longer wavelengths was also very influential. Indeed, the prime characteristic of the MMA was the ability to obtain rapid high-quality images at 230GHz, that is, the MMA was to be a millimeter version of the VLA. The science targets of the MMA included the same broad range of topics seen at the VLA: (NSF) in July, 1990, called for an array of 40 antennas of 8-meter diameter, with four receiver bands covering the atmospheric windows from 30-350 GHz, configurable in four arrays of size 70-3000 m. The proposal discussed two possible sites for the MMA, both in the southwestern United States. Studies of the atmospheric transparency and phase stability at these sites led to similar studies on Mauna Kea, in Hawaii. Extensive atmospheric monitoring was also conducted there.

Concerns with the limited size of the area available to the MMA on Mauna Kea and with potential environmental problems prompted a search for potential sites in Chile. From April 1994, many highelevation sites were visited. Finally, the site retained for the MMA was formally proposed in 1996: it was the Chajnantor plateau.

This effort culminated in the signing of the ALMA Agreement on February 25, 2003, between the North American and European parties. More than 14 government agencies in Chile were involved in the negotiations.

Assuming all three partners are able to meet their commitments, it was decided that the final project would be cost-shared 37.5% / 37.5% / 25% between North America, Europe, and Japan, respectively. The observing time, after a 10% share for Chile, would be shared accordingly.

ALMA is an instrument that consist of a giant array of 12-m antennas with baselines up to 16 km, and an additional compact array of 7-m and 12-m antennas to greatly enhance ALMA's ability to image extended targets, located on the Chajnantor plateau at 5000m altitude. Initially, it will observe at wavelengths in the range 3 mm to 400 m (84 to 720 GHz). The antennas can be moved around, in order to form arrays with different distributions of baseline lengths. More extended arrays will give high spatial resolution, more compact arrays give better sensitivity for extended sources. In addition to the array of 12-m antennas, there is the Atacama Compact Array (ACA), consisting of twelve 7-m antennas and four 12-m antennas. This array will mostly stay in a fixed configuration and is used to image large scale structures that are not well sampled by the ALMA 12-m array.

The design of ALMA is driven by three key science goals:

- The ability to detect spectral line emission from CO in a normal galaxy like the Milky Way at a redshift of $z = 3$, in less than 24 hours
- The ability to image the gas kinematics in protostars and in protoplanetary disks around young Sun-like stars in the nearest molecular clouds ($150pc$)
- The ability to provide precise high dynamic range images at an angular resolution of $0.1arcsec$

ALMA delivers data cubes, of which the third axis is frequency. In this sense, the final data products are very much like that of an integral field unit with up to a million Spectral Pixels.

Interferometry

In order to obtain images, the raw visibility data need to be Fourier transformed. When ALMA is in full operations, this imaging step, as well as various calibration steps, will be done in the data reduction pipeline. Thus, fully calibrated data cubes will be delivered to the user. However, the imaging (and subsequent deconvolution) is a non-unique procedure, so users may want to redo these steps to optimize the data products for their scientific objectives. The Common Astronomy Software Applications package (CASA) has been developed for this purpose.

Observing frequencies

The frequency range available to ALMA is divided into different receiver bands. Data can only be taken in one band at a time. These bands range from band 3, starting at$84GHz$, to band 10, ending at $950GHz$. For comparison, a frequency of $300GHz$ translates to a wavelength of approximately $1mm$. Band 10 ($869GHz$) is planned. Two more bands (band 1 around $40GHz$ and band 2 around $80GHz$), might be added in the future. Initially, only six ALMA antennas will be equipped with band 5 receivers ($187GHz$).

Field of view

The FWHM of the ALMA primary beam is 21" at $300GHz$, and scales linearly with wavelength (diffraction limit of a single 12-m antenna, as opposed to that of the whole array). To achieve uniform sensitivity over a field larger than about a few arcsec, or to image larger regions than the primary beam, mosaicking is required, which is a standard observing mode for ALMA. If you plan to use mosaicking, individual pointings should be separated by 1/2 the primary beam FWHM to achieve Nyquist sampling.

Spatial resolution

The spatial resolution of ALMA depends on the observing frequency and the maximum baseline of the array, following the 1.2 x lambda/D scaling. In the most compact configurations ($150m$), resolutions range from 0.7" at $675GHz$ to 4.8" at $110GHz$. In the most extended configuration (16 km in the completed array), the resolutions range from 6 mas at $675GHz$ to 37 mas at $110GHz$. These numbers refer to the FWHM of the synthesized beam (point spread function), which is the inverse Fourier transform of a (weighted) u-v sampling distribution. The resolution in arcsec can be approximated as: FWHM (") = 76 max_baseline (km) frequency (GHz).

Array configurations

The ALMA 12-m array will cycle from its most compact configuration, with maximum baselines of 150 m, to its most extended configuration, with maximum baselines of 16 km (when completed), and back. The Atacama Compact Array (ACA) will have two configurations, one of which is a north-south extension to provide a better beam shape for far-north/far-south targets. Note that during the Early Science phase, the available configurations are restricted. See 'Capabilities' for conditions applying to the current Cycle.

Spectral resolution

ALMA can deliver data cubes with up to 7680 frequency channels (spectral resolution elements). The width of these channels can range between $3.8kHz$ and $15.6MHz$, but the total bandwidth cannot exceed $8GHz$. At an observing frequency of $110GHz$, the highest spectral resolution implies a velocity resolution of $0.01km/s$, or $R = 30,000,000$. At $110GHz$, a velocity resolution of 1 km/s requires channel widths of $0.37MHz$.

Sensitivity

For an interferometer, the noise level in the resulting data cubes (expressed in mJy) scales roughly as $S = (N * (Np * \nu * \tau)1/2) - 1$, where N is the number of antennas, Np is the number of polarizations, is the available bandwidth and is the observing time. For continuum observations, $\nu = 7.5GHz$, for spectral line observations, is the channel width. The ALMA Sensitivity Calculator can be used to estimate noise levels or required integration times to reach a desired noise level.

2.1.2 ALMA Science Archive

As stated in [8], the purpose of the ALMA archive is to provide services for:

- Persistent archiving and retrieval for observacional data.
- Observaction descrpitors.
- Datacubes produced by pipeline.
- Technical and environmental data.

And the key-point of the conceptual design of the ALMA Archive is to guarantee that three ALMA Regional Centres (ARCs) hold an identical copy of the archive at the Joint ALMA Observatory (JAO) in Santiago.

Relational denormalized database.

ALMA frontend archive is optimized for storage and preservation, not for data query and retrieval. ASA database is inspired from ObsCore, RADAMS and Hubble Legacy Archive plus Virtual Observatory Software:

- openCADC, which is used for database access and VO access protocol
- VOView, for Web components

2.2 Square Kilometer Array

Thousands of linked radio wave receptors will be located in Australia and in Southern Africa. Combining the signals from the antennas in each region will create a telescope with a collecting area equivalent to a dish with an area of about one square kilometre.

The Square Kilometer Array (SKA) will address fundamental unanswered questions about our Universe including how the first stars and galaxies formed after the Big Bang, how galaxies have evolved since then, the role of magnetism in the cosmos, the nature of gravity, and the search for life beyond Earth.

The SKA is a global science and engineering project led by the SKA Organisation, a not-for-profit company with its headquarters at Jodrell Bank Observatory, near Manchester, UK.

An array of dish receptors will extend into eight African countries from a central core region in the Karoo desert of South Africa. A further array of mid frequency aperture arrays will also be built in the Karoo. A smaller array of dish receptors and an array of low frequency aperture arrays will be located in the Murchison Radio-astronomy Observatory in Western Australia.

Construction is scheduled to start in 2016.

Figure 2.2: Artist's impression of the SKA dishes. Credit: SKA Organisation/TD-P/DRAO/Swinburne Astronomy Productions

The recent launch of the Murchison Widefield Array (MWA) a radio telescope based in Western Australia's Mid West - marked the start of an impressive flow of astronomical data that will be stored in the iVEC-managed Pawsey Center in Kensington for later use by researchers around the world.

According to Professor Andreas Wicenec, from The University of Western Australia node of the International Centre for Radio Astronomy Research (ICRAR), SKA has "now have more than 400 megabytes per second of MWA data streaming along the National Broadband Network from the desert 800 km away".

The Murchison Widefield Array is the first Square Kilometre Array precursor to enter full operations, generating a vast torrent of information that needs to be stored for later retrieval by researchers.

According to Proffesor Wicenec, "To store the Big Data the MWA produces, youd need almost three 1 TB hard drives every two hours'. The technical challenge isnt just in saving the observations but how you then distribute them to astronomers from the MWA team in far-flung places so they can start using it".

There are currently two links between the data stores in Perth and MWA researchers at the Massachusetts Institute of Technology (MIT) in the United States and the Victoria University of Wellington in New Zealand. A future link to India another MWA partner will also be created.

The data are not obviously intented to be fully available for everybody at everytime: for instance, MIT researchers are interested in the early universe so filtering techniques to control what data is copied from the Pawsey Center archive to the MIT machines are used. By 2013, more than 150 TB of data had been transferred automatically to the MIT store, with a stream of up to 4 TB a day increasing that value.

MWA is producing so much information that it would be impossible to manually decide where to send what, which is where a sophisticated archiving system the open-source Next Generation Archive System (NGAS) comes in. NGAS was initially developed by Professor Wicenec at the European Southern Observatory (ESO) and later modified by the ICRAR team to meet the MWA data challenge.

The NGAS operating mode is very simple, one simply asks the system for what he/she wants and it either provides it from the local store or retrieves it from the full archive back in Perth through a highly efficient dataflow management system.

About half of all MWA computing occurs on site in the Murchison, where signals from radio telescope antennas are combined and processed in a powerful system of computers called a correlator. Whats left to do in Perth is produce images, and manage storage and distribution by the archive system so MWA astronomers can analyze the collected data. Data travels down a dedicated 10 gigabit per second connection between the Murchison Radio-astronomy Observatory (MRO) and Geraldton, and the trip to Perth is completed on Australias new high-speed National Broadband Network.

The MWA will store about 3 Petabytes at the Pawsey Center each year. Another section of the Pawsey Center will be a supercomputing facility that includes computing for Australias other SKA precursor, CSIROs Australian Square Kilometre Array Pathfinder (ASKAP), and projects from geoscience and other computationally intensive fields.

To sum up, some relevants figures and facts about SKA:

- The data collected by the SKA in a single day would take nearly two million years to playback on an ipod.
- The SKA central computer will have the processing power of about one hundred million PCs.
- The SKA will use enough optical fibre to wrap twice around the Earth!
- The dishes of the SKA will produce 10 times the global internet traffic.
- The aperture arrays in the SKA could produce more than 100 times the global internet traffic.
- The SKA will generate enough raw data to fill 15 million 64 GB iPods every day!
- The SKA supercomputer will perform 1018 operations per second equivalent to the number of stars in three million Milky Way galaxies in order to process all the data that the SKA will produce.
- The SKA will be so sensitive that it will be able to detect an airport radar on a planet 50 light years away.
- The SKA will contain thousands of antennas with a combined collecting area of about one square kilometre (thats 1 000 000 square metres!).
- Analysts estimate the London Olympics was the most data-heavy yet with some 60 Gbytes, the equivalent of 3,000 photographs, travelling across the

network in the Olympic Park every second. This however is only equivalent to the data rate from about half a low frequency aperture array station in SKA phase one.

Chapter 3

The Virtual Observatory

The International Virtual Observatory Alliance (IVOA) was formed in June 2002 with a mission to "facilitate the international coordination and collaboration necessary for the development and deployment of the tools, systems and organizational structures necessary to enable the international utilization of astronomical archives as an integrated and interoperating virtual observatory." The IVOA now comprises 20 VO programs from Argentina, Armenia, Australia, Brazil, Canada, China, Europe, France, Germany, Hungary, India, Italy, Japan, Russia, Spain, the United Kingdom, Ukraine, and the United States and inter-governmental organizations (ESA and ESO). Membership is open to other national and international programs according to the IVOA Guidelines for Participation.

The IVOA focuses on the development of standards and encourages their implementation for the benefit of the worldwide astronomical community. Working Groups are constituted with cross-program membership in those areas where key interoperability standards and technologies have to be defined and agreed upon. The Working Groups develop standards using a process modeled after the World Wide Web Consortium, in which Working Drafts progress to Proposed Recommendations and finally to Recommendations. Recommendations may ultimately be endorsed by the Virtual Observatory Working Group of Commission 5 (Astronomical Data) of the International Astronomical Union. The IVOA also has Interest Groups that discuss experiences using VO technologies and provide feedback to the Working Groups.

Ad-hoc and permanent committees deal with specific scientific and procedural topics. Interaction with other scientific disciplines interested in data inter-operability is also pursued through dedicated Liaison Groups.

Senior representatives from each national and international member VO program form the IVOA Executive Committee. A chair is chosen from among the representatives and serves an eighteen-month term, normally preceded by an eighteen-month term as deputy chair. The Executive Committee meets 3-4 times a year (also by teleconference) to discuss goals, priorities, and strategies. Executive Committee members represent their respective programs and are expected to be in a position to commit resources targeted at the achievement of common goals. Decisions by the IVOA Executive Committee are reached by consensus.

The IVOA holds two Interoperability Workshops each year typically in May and October. These meetings are opportunities for the IVOA Groups and Committees to have face-to-face discussions.

3.1 ObsTAP

In 2011 IVOA proposed a new recommendation: Observation Data Model Core Components and its Implementation in the Table Access Protocol. That document was intended to be a description of the interface which integrated the data modeling and data access aspects in a single service:

$\boxed{\text{ObsCore data model + Table Access Protocol = ObsTAP}}$

3.2 FITS format

Flexible Image Transport System (FITS) is an open standard defining a digital file format useful for storage, transmission and processing of scientific and other images. FITS is the most commonly used digital file format in astronomy. Unlike many image formats, FITS is designed specifically for scientific data and hence includes many provisions for describing photometric and spatial calibration information, together with image origin metadata.

The FITS format was first standardized in 1981; it has evolved gradually since then, and the most recent version (3.0) was standardized in 2008. FITS was designed with an eye towards long-term archival storage, and the maxim once FITS, always FITS represents the requirement that developments to the format must be backwards compatible.

A major feature of the FITS format is that image metadata is stored in a human-readable ASCII header, so that an interested user can examine the headers to investigate a file of unknown provenance. The information in the header is designed to calculate the byte offset of some information in the subsequent data unit to support direct access to the data cells. Each FITS file consists of one or more headers containing ASCII card images (80 character fixed-length strings) that carry keyword/value pairs, interleaved between data blocks. The keyword/value pairs provide information such as size, origin, coordinates, binary data format, free-form comments, history of the data, and anything else the creator desires: while many keywords are reserved for FITS use, the standard allows arbitrary use of the rest of the name-space.

FITS is also often used to store non-image data, such as spectra, photon lists, data cubes, or even structured data such as multi-table databases. A FITS file may contain several extensions, and each of these may contain a data object. For example, it is possible to store x-ray and infrared exposures in the same file.

FITS support is available in a variety of programming languages that are used for scientific work, including C, C++, C#, Fortran, IGOR Pro, IDL, Java, LabVIEW, Mathematica, MatLab, Perl, PDL, Python, R, and Tcl. The FITS Support Office at NASA/GSFC maintains a list of libraries and platforms that currently support FITS.

Image processing programs such as ImageJ, GIMP, Photoshop, XnView and IrfanView can generally read simple FITS images, but frequently cannot interpret

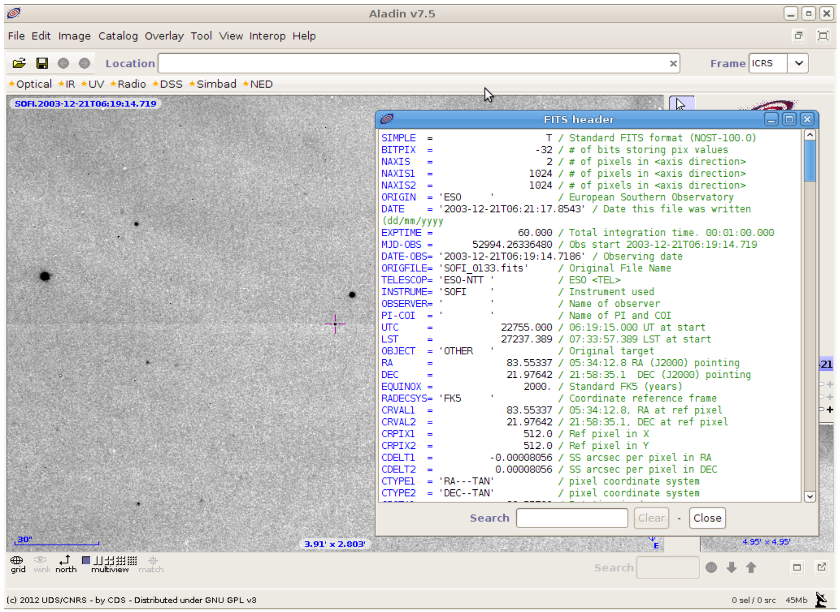


Figure 3.1: Viewing FITS header in Aladin

more complex tables and databases. Scientific teams frequently write their own code to interact with their FITS data, using the tools available in their language of choice. The FITS Liberator software is used by imaging scientists at the European Space Agency, the European Southern Observatory and NASA. The SAOImage DS9 Astronomical Data Visualization Application is available for many OSs, and handles images and headers.

3.3 OpenCADC

Esto es una prueba

Chapter 4

Data storage and access

Typically, modern relational databases have shown little efficiency in certain applications using intensive data, like indexing of a large number of documents, sites rendering with high traffic, and streaming sites. Typical RDBMS implementations are tuned either for small but frequent reads and writes or a large set of transactions that have few write accesses. On the other hand NoSQL can serve load lots of reads and writes.

We will start making a short overview of the relational model using Postgresql RDMS. Then we move to the NoSQL alternative, through Mongo database.

4.1 Relational Database Management System

ToDo

4.1.1 Postgresql

PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most SQL:2008 data types, including INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, among others, and exceptional documentation.

An enterprise class database, PostgreSQL boasts sophisticated features such as Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions (savepoints), online/hot backups, a sophisticated query planner/optimizer, and write ahead logging for fault tolerance. It supports international character sets, multibyte character encodings, Unicode, and it is locale-aware for sorting, case-sensitivity, and formatting. It is highly scalable both in the sheer quantity of data it can manage and in the number of concurrent users it can accommodate. There are active PostgreSQL systems in production environments that manage in excess of 4 terabytes of data. Some general PostgreSQL limits are included in the table below.

PostgreSQL runs stored procedures in more than a dozen programming languages, including Java, Perl, Python, Ruby, Tcl, C/C++, and its own PL/pgSQL, which is similar to Oracle's PL/SQL. Included with its standard function library are hundreds of built-in functions that range from basic math and string operations to cryptography and Oracle compatibility. Triggers and stored procedures can be written in C and loaded into the database as a library, allowing great flexibility in extending its capabilities. Similarly, PostgreSQL includes a framework that allows developers to define and create their own custom data types along with supporting functions and operators that define their behavior. As a result, a host of advanced data types have been created that range from geometric and spatial primitives to network addresses to even ISBN/ISSN (International Standard Book Number/International Standard Serial Number) data types, all of which can be optionally added to the system.

Just as there are many procedure languages supported by PostgreSQL, there are also many library interfaces as well, allowing various languages both compiled and interpreted to interface with PostgreSQL. There are interfaces for Java (JDBC), ODBC, Perl, Python, Ruby, C, C++, PHP, Lisp, Scheme, and Qt just to name a few.

Best of all, PostgreSQL's source code is available under a liberal open source license: the PostgreSQL License. This license gives you the freedom to use, modify and distribute PostgreSQL in any form you like, open or closed source. Any modifications, enhancements, or changes you make are yours to do with as you please. As such, PostgreSQL is not only a powerful database system capable of running the enterprise, it is a development platform upon which to develop in-house, web, or commercial software products that require a capable RDBMS.

Limit	**Value**
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250 - 1600 depending on column types
Maximum Indexes per Table	Unlimited

4.2 NoSQL

NoSQL implementations used in the real world include 3TB Digg green markers (indicated to highlight the stories voted by others in the social network), the 6 TB of "ENSEMBLE" European Commission database used in comparing models and air quality, and the 50 TB of Facebook inbox search.

NoSQL architectures often provide limited consistency, such as events or transactional consistency restricted to only data items. Some systems, however, provide all guarantees offered by ACID systems by adding an intermediate layer. There are

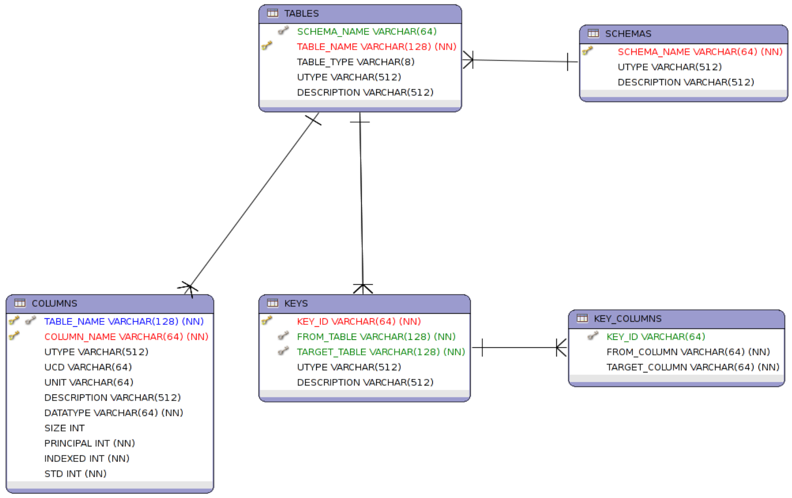


Figure 4.1: Entity-Relationship Diagram for VO Tap Schema

two systems that have been deployed and provide for storage of snapshot isolation column: Google Percolator (based on BigTable system) and Hbase transactional system developed by the University of Waterloo. These systems use similar concepts in order to achieve distributed multiple rows ACID transactions with snapshot isolation guarantees for the underlying storage system in that column, wit no extra overhead in data management, no system deployment middleware or any maintenance introduced by middleware layer.

Quite NoSQL systems employ a distributed architecture, maintaining data redundantly on multiple servers, often using distributed hash table. Thus, the system may actually escale adding more servers, and thus a server failure may be tolerated.

There are different NoSQL DBs for different projects:

- Document oriented
  - CouchDB
  - MongoDB

  - RavenDB
  - BaseX
  - djondb
  - eXist
  - SimpleDB
  - IBM Lotus Domino
  - Terrastore

- Graph oriented
  - Neo4j
  - DEX
  - AllegroGraph
  - OrientDB
  - InfiniteGraph
  - Sones GraphDB
  - InfoGrid
  - HyperGraphDB

- Key-value oriented
  - Cassandra
  - BigTable
  - Dynamo (Amazon)
  - MongoDB
  - Project Voldemort (LinkedIn)
  - Riak
  - Redis

- Multivalue
  - OpenQM
  - Extensible storage engine
- Object Oriented
  - Zope Object Database
  - db4o
  - GemStone S
  - Objectivity/DB
- Tabular
  - HBase
  - BigTable
  - LevelDB (BigTable open version)
  - Hypertable

They run on clusters of inexpensive machines.

4.2.1 MongoDB

MongoDB (from "humongous") is an open source document-oriented database system developed and supported by 10gen. It is part of the NoSQL family of database systems. Instead of storing data in tables as is done in a "classical" relational database, MongoDB stores structured data as JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster.

10gen began Development of MongoDB in October 2007 and was not created to be just another database that tries to do everything for everyone. Instead, MongoDB was created to work with documents rather than rows, was extremely fast, massively scalable, and easy to use. In order to accomplish this, some features were excluded, namely support for transactions.

Main features

- Ad hoc queries MongoDB supports search by field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions.

- Indexing Any field in a MongoDB document can be indexed (indices in MongoDB are conceptually similar to those in RDBMSes). Secondary indices are also available.

- Replication MongoDB supports master-slave replication. A master can perform reads and writes. A slave copies data from the master and can only be used for reads or backup (not writes). The slaves have the ability to select a new master if the current one goes down.

- Load balancing MongoDB scales horizontally using sharding.[9] The developer chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more slaves.) MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure. Automatic configuration is easy to deploy and new machines can be added to a running database.

- File storage MongoDB could be used as a file system, taking advantage of load balancing and data replication features over multiple machines for storing files. This function, called GridFS,[10] is included with MongoDB drivers and available with no difficulty for development languages (see "Language Support" for a list of supported languages). MongoDB exposes functions for file manipulation and content to developers. GridFS is used, for example, in plugins for NGINX.[11] and lighttpd[12] In a multi-machine MongoDB system, files can be distributed and copied multiple times between machines transparently, thus effectively creating a load balanced and fault tolerant system.

- Aggregation MapReduce can be used for batch processing of data and aggregation operations. The aggregation framework enables users to obtain the kind of results for which the SQL GROUP BY clause is used.
- Server-side JavaScript execution JavaScript can be used in queries, aggregation functions (such as MapReduce), are sent directly to the database to be executed.
- Capped collections MongoDB supports fixed-size collections called capped collections. This type of collection maintains insertion order and, once the specified size has been reached, behaves like a circular queue.

  Once we have seen the main features of MongoDB, we can move on to the language itself.

The basics

We must know four concepts to dig into MongoDB's world:

- Database: this concept is much like the RDBM counterpart.
- Collection: we can see a collection and a table as the same thing.
- Document: its equivalent in RDBM is the row, and a document is made up of fields.
- Field: is a lot like a column.

4.2.2 Advantages and uncertainties of using a NoSQL solution

- Elastic scaling

  For years, database administrators have relied on scale up buying bigger servers as database load increases rather than scale out distributing the database

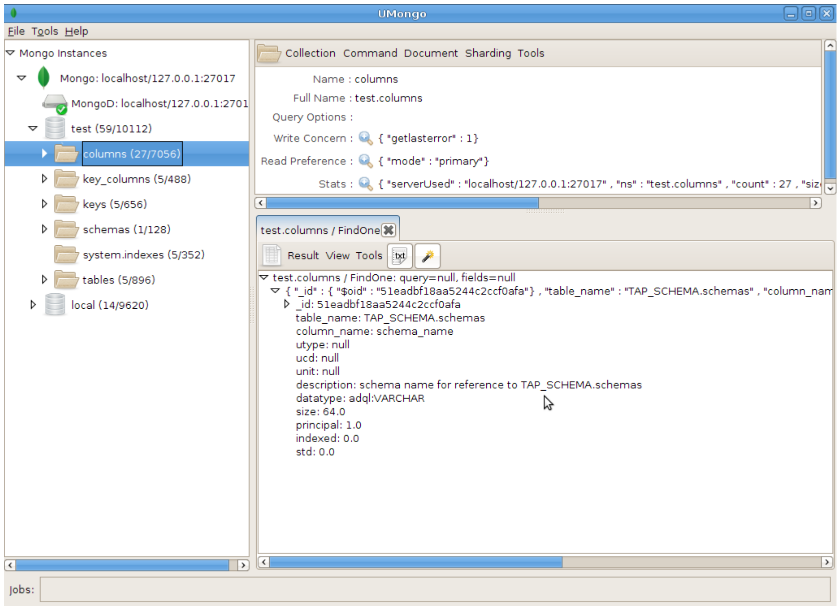


Figure 4.2: Tree View for Tap Schema in MongoVue

across multiple hosts as load increases. However, as transaction rates and availability requirements increase, and as databases move into the cloud or onto virtualized environments, the economic advantages of scaling out on commodity hardware become irresistible.

RDBMS might not scale out easily on commodity clusters, but the new breed of NoSQL databases are designed to expand transparently to take advantage of new nodes, and theyre usually designed with low-cost commodity hardware in mind.

- Big data

  Just as transaction rates have grown out of recognition over the last decade, the volumes of data that are being stored also have increased massively. OReilly has cleverly called this the industrial revolution of data. RDBMS capacity has been growing to match these increases, but as with transaction rates, the constraints of data volumes that can be practically managed by a single RDBMS are becoming intolerable for some enterprises. Today, the volumes of big data that can be handled by NoSQL systems, such as Hadoop, outstrip what can be handled by the biggest RDBMS.

- No need for DBAs

  Despite the many manageability improvements claimed by RDBMS vendors over the years, high-end RDBMS systems can be maintained only with the assistance of expensive, highly trained DBAs. DBAs are intimately involved in the design, installation, and ongoing tuning of high-end RDBMS systems.

  NoSQL databases are generally designed from the ground up to require less management: automatic repair, data distribution, and simpler data models lead to lower administration and tuning requirements in theory. In practice, its likely that rumors of the DBAs death have been slightly exaggerated. Someone will always be accountable for the performance and availability of any mission-critical data store.

- Economics

  NoSQL databases typically use clusters of cheap commodity servers to manage the exploding data, while RDBMS tends to rely on expensive proprietary servers and storage systems. The result is that the cost per gigabyte or transaction/second for NoSQL can be many times less than the cost for RDBMS, allowing you to store and process more data at a much lower price point.

- Flexible data models

  Change management is a big headache for large production RDBMS. Even minor changes to the data model of an RDBMS have to be carefully managed and may necessitate downtime or reduced service levels.

  NoSQL databases have far more relaxed or even nonexistent data model restrictions. NoSQL Key Value stores and document databases allow the application to store virtually any structure it wants in a data element. Even the more rigidly defined BigTable-based NoSQL databases (Cassandra, HBase) typically allow new columns to be created without too much fuss.

  The result is that application changes and database schema changes do not have to be managed as one complicated change unit. In theory, this will allow applications to iterate faster, though,clearly, there can be undesirable side effects if the application fails to manage data integrity.

NoSQL systems have generated a lot of enthusiasm but there are still a lot of questions about its future:

- Maturity

  RDBMS systems have been around for a long time. NoSQL advocates will argue that their advancing age is a sign of their obsolescence, but for most CIOs, the maturity of the RDBMS is reassuring. For the most part, RDBMS systems are stable and richly functional. In comparison, most NoSQL alternatives are in pre-production versions with many key features yet to be implemented.

  Living on the technological leading edge is an exciting prospect for many developers, but enterprises should approach it with extreme caution.

- Support

  Enterprises want the reassurance that if a key system fails, they will be able to get timely and competent support. All RDBMS vendors go to great lengths to provide a high level of enterprise support.

  In contrast, most NoSQL systems are open source projects, and although there are usually one or more firms offering support for each NoSQL database, these companies often are small start-ups without the global reach, support resources, or credibility of an Oracle, Microsoft, or IBM.

- Analytics and business intelligence

  NoSQL databases have evolved to meet the scaling demands of modern Web 2.0 applications. Consequently, most of their feature set is oriented toward the demands of these applications. However, data in an application has value to the business that goes beyond the insert-read-update-delete cycle of a typical Web application. Businesses mine information in corporate databases to improve their efficiency and competitiveness, and business intelligence (BI) is a key IT issue for all medium to large companies.

NoSQL databases offer few facilities for ad-hoc query and analysis. Even a simple query requires significant programming expertise, and commonly used BI tools do not provide connectivity to NoSQL.

Some relief is provided by the emergence of solutions such as HIVE or PIG, which can provide easier access to data held in Hadoop clusters and perhaps eventually, other NoSQL databases. Quest Software has developed a product Toad for Cloud Databases that can provide ad-hoc query capabilities to a variety of NoSQL databases.

- Administration

  The design goals for NoSQL may be to provide a zero-admin solution, but the current reality falls well short of that goal. NoSQL today requires a lot of skill to install and a lot of effort to maintain.

- Expertise

  There are literally millions of developers throughout the world, and in every business segment, who are familiar with RDBMS concepts and programming. In contrast, almost every NoSQL developer is in a learning mode. This situation will address naturally over time, but for now, it is by far easier to find experienced RDBMS programmers or administrators than NoSQL experts.

Chapter 5

ASA with MongoDB

Esto es una prueba

Chapter 6

Successful experiences

Chapter 7

Conclusions and future work

Conclusions and future work.

Appendix A

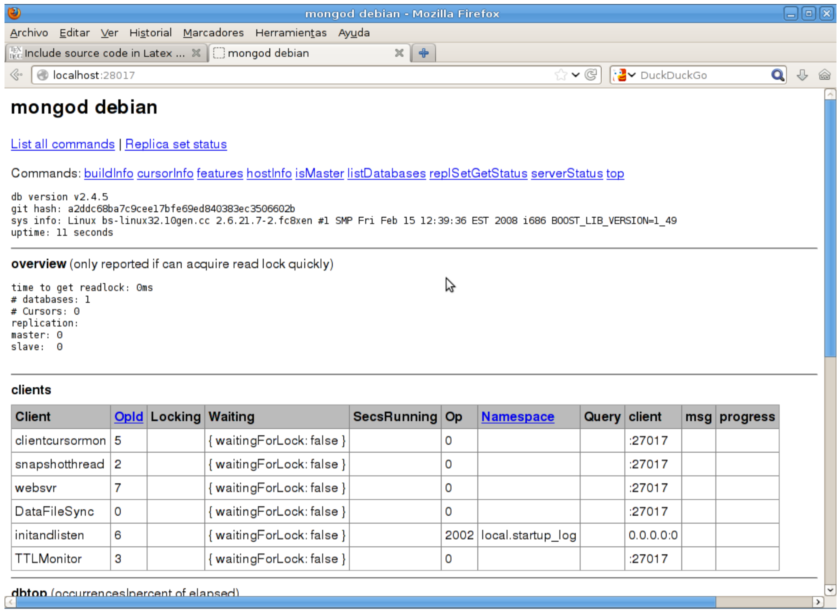
Getting and installing MongoDB

- Go to http://www.mongodb.org/downloads and select our OS version. Download it and decompress.

- Supposing we are in a Linux box and that we are using the latest release (in June 2013) execute the following command:

```
./mongod --rest --dbpath /opt/mongodb-linux-i686-2.4.5/bin/data/db/
```

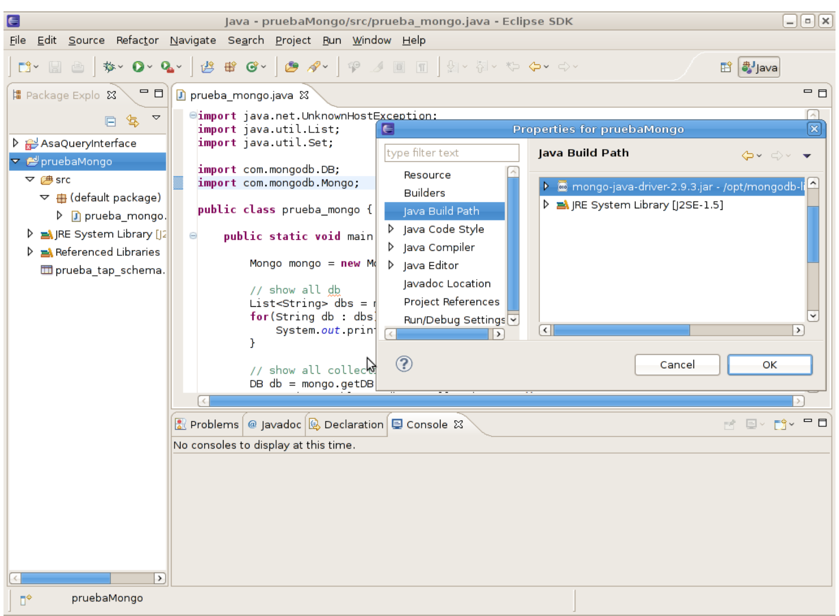
- The server is now listening and waiting for connections, by default, in port 27017. As we have used the *–rest* modifier, we can point our browser to http://localhost:28017 for http diagnostic access.


mongod debian - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
Include source code in Latex ...
mongod debian
localhost:28017
DuckDuckGo
mongod debian
List all commands | Replica set status
Commands: buildInfo cursorInfo features hostInfo isMaster listDatabases replSetGetStatus serverStatus top
db version v2.4.5
git hash: a2ddc68ba7c9cee17bfe69ed840383ec3506602b
sys info: Linux bs-linux32.10gen.cc 2.6.21.7-2.fc8xen #1 SMP Fri Feb 15 12:39:36 EST 2008 i686 BOOST_LIB_VERSION=1_49
uptime: 11 seconds
overview (only reported if can acquire read lock quickly)
time to get readlock: 0ms
databases: 1
Cursors: 0
replication:
master: 0
slave: 0
clients
Client | OpId | Locking | Waiting | SecsRunning | Op | Namespace | Query | client | msg | progress
clientcursormon | 5 | { waitingForLock: false } | 0 | :27017
snapshotthread | 2 | { waitingForLock: false } | 0 | :27017
websvr | 7 | { waitingForLock: false } | 0 | :27017
DataFileSync | 0 | { waitingForLock: false } | 0 | :27017
initandlisten | 6 | { waitingForLock: false } | 2002 | local.startup_log | 0.0.0.0:0
TTLMonitor | 3 | { waitingForLock: false } | 0 | :27017
dbtop (occurrences|percent of elapsed)

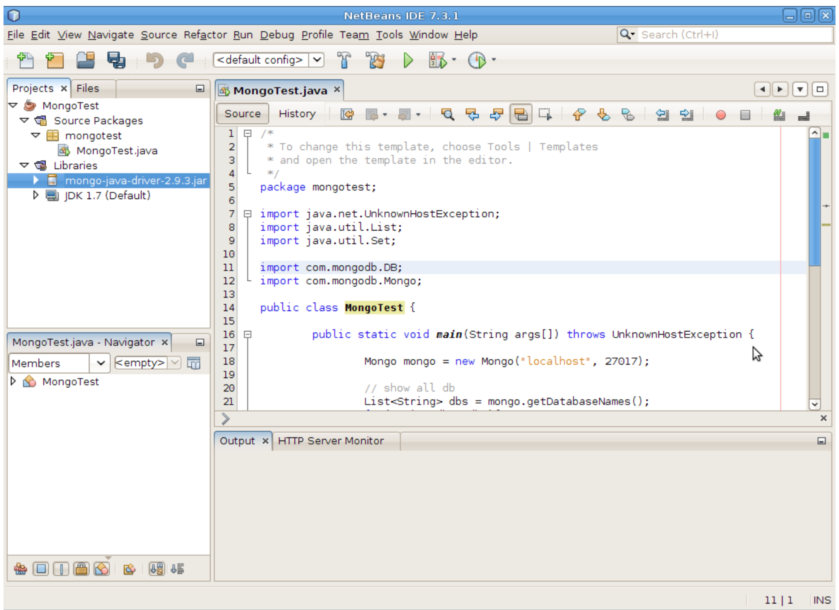
Appendix B

Configuring Eclipse and NetBeans for Java/Mongo development

- Go to http://central.maven.org/maven2/org/mongodb/mongo-java-driver/ and choose the right version.
- In Eclipse, just start a new project **File - New - Java Project**
- Right-click in **Properties** and then **Java Build Path - Libraries - Add External JARs**



- Import MongoDB methods, classed and interfaces as needed.
- In NetBeans, start a new project **File - New Project - Java Application**
- Right-click in **Libraries** and then **Add JAR/Folder**



- Import MongoDB methods, classed and interfaces as needed.

Bibliography

[1] Inc 10gen. The mongodb 2.4 manual. http://docs.mongodb.org/manual/, 2011-2013. [Online; accessed 01-December-2012].

[2] Kristina Chodorow. *MongoDB: The Definitive Guide*. O'Reilly, 2010.

[3] Kristina Chodorow. *50 Tips and Tricks for MongoDB Developers*. O'Reilly, 2011.

[4] Juan de Dios Santander Vela. Bases de datos multimedia para radioastronomía: Radams y dss-63. http://www.iaa.es/~jdsant/DEA-BBDDMultimediaParaRadioastronomia-RADAMS%2CDSS63.pdf, 2006. [Online; accessed 01-May-2013].

[5] Juan de Dios Santander Vela et al. The alma science archive: Design. ftp://ftp.eso.org/projects/adass/posters/P147.pdf, 2011. [Online; accessed 27-April-2013].

[6] Juan de Dios Santander Vela et al. The alma science archive: Implementation. ftp://ftp.eso.org/projects/adass/posters/P133.pdf, 2011. [Online; accessed 07-February-2013].

[7] Abraham Silberschatz et al. *Database System Concepts Sixth Edition*. McGraw-Hill, 2010.

[8] Sandra Etoka. A first look at the alma science archive. http://www.eso.org/sci/facilities/alma/meetings/gar-sep07/pdf/Etoka.pdf, 2012. [Online; accessed 21-March-2013].

[9] Centre for Astrophysics and Supercomputing. Scientific computing and visualisation. http://astronomy.swin.edu.au/scivis/, 2013. [Online; accessed 30-January-2013].

[10] Centre for Astrophysics and Supercomputing. Accelerating astrophysics research with gpu supercomputing. http://www.sgi.com/pdfs/4401.pdf, 2013. [Online; accessed 30-January-2013].

[11] Nature Vol 455 — Issue no. 7209. Big data: science in the petabyte era. http://www.nature.com/nature/journal/v455/n7209/edsumm/e080904-01.html, 2008. [Online; accessed 14-February-2011].

[12] IBM Research. Square kilometer array: Ultimate big data challenge. http://asmarterplanet.com/blog/2013/03/the-square-kilometer-array-the-world%E2%80%99s-ultimate-big-data-challenge.html, 2013. [Online; accessed 04-July-2012].

[13] John Richer. Alma, synthesis imaging and the astro-grid. http://thames.cs.rhul.ac.uk/~fionn/astro-grid-papers/john-richer.pdf, 2001. [Online; accessed 22-November-2012].

[14] skatelescope.org. The age of astronomy big data is already here. http://www.skatelescope.org/news/pawsey-centre/, 2013. [Online; accessed 09-July-2012].