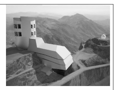
Petasky: some query optimization challenges related to management of scientific data in the field of cosmology

Farouk Toumani

CNRS, Blaise Pascal University, Aubiere, France

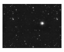
Abstract

In modern astronomy, with a massive set of digital information of unprecedented volumes being collected from measurements and computational models, it becomes more and more difficult to manage and analyse/mine large data repositories. LSST (Large Synoptic Survey Telescope, http://www.lsst.org/), which targets the construction of a telescope of a new generation, is an example of a project where data management and analysis is recognized as very challenging as more than 30 TB (TeraBytes) of complex data (3.2 Gigapixel images, uncertain data, multi-scale data) must be processed and stored each night to produce the largest non-proprietary data set in the world. This talk describes the work achieved in the context of the Petasky project to identify LSST requirements in terms of data management and the underlying challenges. It also reports on first results regarding the evaluation of emerging bigdata management technologies in this context and identifies some relevant research directions.

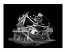


PetaSky

Some query optimization challenges related to management of scientific data in the field of cosmology
http://com.isima.fr/Petasky



F. Toumani, LIMOS, Clermont-Ferrand



challenges in astro- and environmental informatics in the Big Data era, 14-16 May, 2014, Szombathely, Hung

(initial) Agenda

- The Petasky project
- Data management issues
- Explored approaches
- Learned lessons and research directions

Main thesis of this talk

How to form a new generation of scientists capable to exploit the new technologies to pursue science goals at an unprecedent scale?

G.Longo (this morning)

It should not be up to the scientists but to the technology (data management system) **to overcome the computing barriers** between them and the data

Main thesis of this talk

This approach has been very successful in business domain but in general not so successful in the scientific domain

It should not be up to the scientists but to the technology (data management system) **to overcome the computing barriers** between them and the data

Petasky: data management challenge

Techniques to build an **efficient** and **easy to use** data access system at a **reasonable cost**

- Specialized Hardware
- Programming
- Ad-hoc optimization

- Commodity machines
- Querying
- Generic system

Consortium

- INS2I
  - LIMOS (UMR CNRS 6158, Clermont-Ferrand)
  - LIRIS (UMR CNRS 5205, Lyon)
- IN2P3
  - LPC (UMR CNRS 6533, Clermont-Ferrand)
  - APC (UMR CNRS 7164, Paris)
  - LAL (UMR CNRS 8607, Paris)
  - Centre de Calcul de l'IN2P3/CNRS (CC-IN2P3)

Science in an exponential world

The availability of very large amounts of data and the ability to efficiently process them is changing the way we do science

- Science paradigms[1]
  1. **Empirical** description of natural phenomena
  2. **Theoretical** science: models and generalization
  3. **Computational** science: simulation of complexe phenomena to validate theories
  4. **Data Intensive science** : collecting and analyzing large amount of data

[1]Jim Gray, eScience Talk at NRC-CSTB meeting Mountain View CA, 11 January 2007.

Science in an exponential world

The availability of very large amounts of data and the ability to efficiently process them is changing the way we do science

- Science paradig
  1. **Emp**
  2. **The**
  3. **Computational** science: ulation of complexe phenomena to validate theories
  4. **Data Intensive science** llecting and analyzing large amount of data

Hypothesize
Design and run experiments
Analyze results

Hypothesize
Look up answers in database

[1]Jim Gray, eScience Talk at NRC-CSTB meeting Mountain View CA, 11 January 2007.

Petasky: scientific challenges

- Management of scientific data in the fields of **cosmology and astrophysics**
  - Large amount of data
  - Complex data (e.g., images, uncertainty, multi-scales...)
  - Heterogeneous formats
  - Various and complex processing (images analysis, reconstruction of trajectories, ad-hoc queries and processings, …)
- Scientific challenges
  - Scalability
  - Data integration
  - Data analysis
  - Visualisation
- Application context : **LSST project**

The LSST project
Large Synoptic Survey telescope

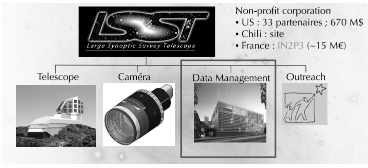
The New Sky

WIDE
A large aperture, wide field survey telescope and 3200 Megapixel camera to image faint astronomical objects across the sky.

FAST
LSST will rapidly scan the sky, charting objects that change or move: from exploding supernovae to potentially hazardous near-Earth asteroids.

DEEP
LSST's images will trace billions of remote galaxies, providing multiple probes of the mysterious dark matter and dark energy.

The LSST



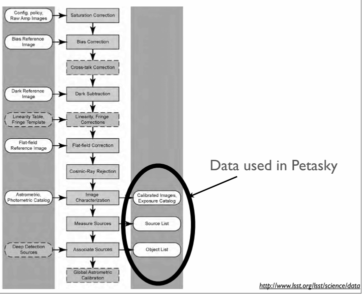
Data management challenges in LSST

*"How much the (LSST) project will tell us about our solar system, the dark energy problem and more, will **depend on how well we can process the information** the telescope and its camera send back to us - an estimated sum of around ten petabytes of data per year."*
(Mari Silbey, *Space: the big data frontier*, http://www.smartplanet.com/blog/thinking-tech/space-the-big-data-frontier/12180)

*"Plans for sharing the data from LSST with the public are **as ambitious as the telescope itself**"*

Anyone with a computer will be able to fly through the Universe, zooming past objects a hundred million times fainter than can be observed with the unaided eye. The LSST project will provide analysis tools to enable both students and the public to participate in the process of scientific discovery.

Data production in LSST



http://www.lsst.org/lsst/science/data

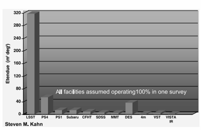
LSST data scales

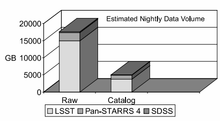
Tens of thousands of billions of photometric observations over tens of billions of objects

- 1-10 Millions events per night
- 3 billions of sources
- 16 TB each 8 hours at a rate of 540 MB/second
- Objects catalog :
  - Relation with **500 attributes, 40 Billions of tuples**
  - **100-200 TB**
- Transients
  - 1-3 PB, Relation with **100 attributes/5000 billions of tuples**
- Image
  - 6 GB/17 seconds
  - **100 PB** final archives of images
- Estimation for the end of the project : **400 000 Billions of tuples** (different versions of data in addition to replication), ≈**60 PB**

LSST scale data sets

A change of scale from TB to PB





Queries per difficulty level
Supported queries

- Retrieve any type of information about a single object (identified by a given objectId), including full time series.

  SELECT * FROM Object JOIN Source USING (objectId) WHERE objectId = 293848594;

  Few seconds
- Retrieve any type of information about a group of objects in a small area of sky, including neighborhood-type queries.

  SELECT * FROM Object WHERE qserv_areaSpec_circle(1.0, 35.0, 5.0/60)

  ≈ **1 hour**
- Analysing light curves across large area.

  SELECT O.objectId, myFunction(S.taiMidPoint, S.psfFlux) FROM Object AS O JOIN Source AS S USING (objectId) WHERE O.varProb > 0.75 GROUP BY O.objectId;

  ≈ **1 day (24h)**
- Analysing light curves of faint objects across large area.

  SELECT O.objectId, myFunction(V.taiMidPoint, FS.flux) FROM Object AS O JOIN ForcedSource AS FS ON (O.objectId = FS.objectId) JOIN Visit AS V ON (FS.visitId = V.visitId);

  ≈ **1 week**

Queries per difficulty level
Expensive/impossible queries

- **Expensive queries**
  - Find objects far away from other objects (for a large number of objects).
    - Question: what is the largest distance we should plan to support for distance-based queries involving (a) small number of objects, (b) all objects on the sky?
  - Sliding window queries: Find all 5 arcmin x 5 arcmin regions with an object density higher than rho
- **Impossible queries**
  - Large size results
    - Select all pairs of stars within 1 arc min of each other in the Milky Way region.
  - Expensive or hidden computation (e.g., Join)
    - Near neighbor query on the Source or ForcedSource table
    - Joining large tables between different LSST data releases
    - Time series analysis of every object
    - Cross-match with very large external catalog (e.g. LSST with SKA)
    - Any non-spatial join on the entire catalog (Object, Source, ForcedSource)
    - Join of Source with ForcedSource

LSST scale data sets

	LSST year 1	LSST year 10
Raw data	6 F	
Archive	19	
Disk (DAC)	16	
DB (baseline)	0,5	
Moore equivalent 2014	1.2	

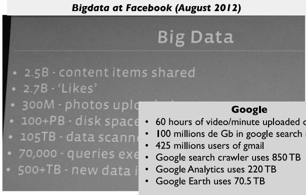
Table	*Size*	*#tuples*	*#attributes*
Object	109 TB	38 B	470
Moving Object	5 GB	6 M	100
Source	3.6 PB	5 T	125
Forced Source	1.1 PB	32 T	7
Difference Image Source	71 TB	200 B	65
CCD Exposure	0.6 TB	17 B	45

12 TB : >> standards DB sizes in DBMSs → **Big**
~6H for a full scan at 600 Mb/s.
110H to index 3TB with MySQL
Beyond the capacity of a centralized system -> need for a distributed system
Moore law is not true ...
Data and bandwidth will continue to grow exponentially but latency will grow sub-linearly or not at all

Data management challenges in LSST

How big is big?

Bigdata at Facebook (August 2012)

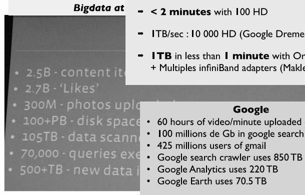


Google
- 60 hours of video/minute uploaded on youtube
- 100 millions de Gb in google search index
- 425 millions users of gmail
- Google search crawler uses 850 TB
- Google Analytics uses 220 TB
- Google Earth uses 70.5 TB

Data management challenges in LSST

How

Bigdata at



- Table scan : ≈**3h** to scan **1 TB**
- Parallelization
  - **< 2 minutes** with 100 HD
  - 1TB/sec : 10 000 HD (Google Dremel)
  - **1TB** in less than **1 minute** with Oracle DBMS : 2 RAC nodes + Multiples infiniBand adapters (Maklee)

Google
- 60 hours of video/minute uploaded on youtube
- 100 millions de Gb in google search index
- 425 millions users of gmail
- Google search crawler uses 850 TB
- Google Analytics uses 220 TB
- Google Earth uses 70.5 TB

Example of a DBMS X Big Data configuration

- A full rack configuration with 18 Sun servers
- A total storage capacity of 648TB
- Every server in the rack has 2 CPUs, each with 8 cores for a total of 288 cores per full rack
- Each server has 64GB memory for a total of 1152GB of memory per full rack.

Petasky: data management challenge

Techniques to build an **efficient** and **easy to use** data access system at a **reasonable cost**

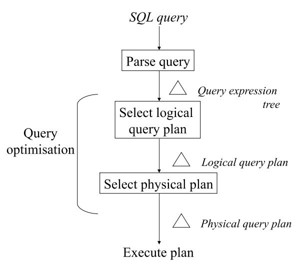
- Specialized Hardware
- Programming
- Ad-hoc optimization

- Commodity machines
- Querying
- Generic system

What makes DB technology successful in business domain?

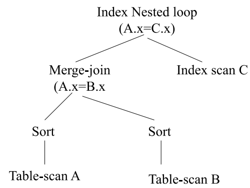
- Abstractions
  - ✓ **Relation** instead of files, blocks, tablespaces, segments, extents, access path
  - ✓ Relational algebra instead of algorithms
- Declarative query language
  - ✓ Express what you want not how to get it
- Optimization
  - ✓ Rather naive techniques but enough for the business world

Example: query compilation



Example: query compilation

Select Name
From Emp, Dep
Where empno=mgr



Space of solutions and associated challenges

! Clearly beyond the capacities of centralized systems

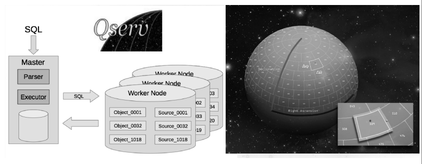
- Distributed and parallel systems
  - ✓ Data distribution
  - ✓ Computation distribution
  - ✓ Failure resilience
- Storage model
  - ✓ row store vs. column store
  - ✓ (sophisticated) Indexes
- Benefit from modern hardware
- Complexity theory and cost models
  - ✓ Standards measures: I/O, data transfer, ..
  - ✓ Cost of coordination

Existing approaches

- Big data approaches
  - ✓ Distributed and parallel systems
    - ➡ MapReduce like approaches (shared nothing architecture)
    - ➡ Parallel DBMS (shared all thing architecture)
    - ➡ Spatial partitioning (QSERV, pour LSST)
  - ✓ Column store DBMSs
    - ➡ Vertica, MonetDB, ...
- Data integration to the rescue
  - ➡ Declarative approach

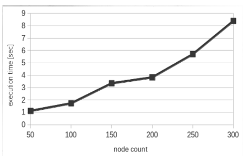
Qserv

- Data partitioning
  - ✓ Horizontal fragmentation of the table Object: $\sigma_{\theta i}$***(Object)***
  - ✓ Derived horizontal fragmentation of the table Source: π**(Source** $\ltimes\sigma_{\theta i}$***(Object))***
- Needs for query orchestration



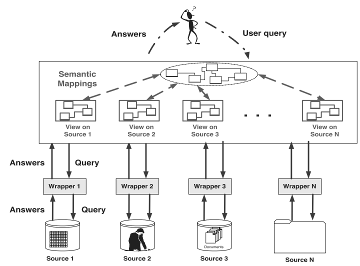
Experiments with QServ

- 300 nodes, 120 GB of storage, 16 GB of RAM
- 15 TB, 3000 chunks → 50 GB/node

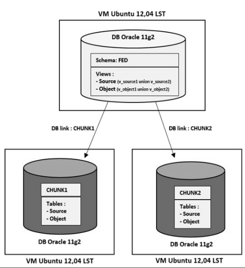


- Limitations
  - ✓ Limited queries
    - distance ~1 arcmin
    - non spatial joins
    - Non-partitionable aggregates
  - ✓ Load balancing
  - ✓ Ad-hoc query rewriting

Data integration at the rescue



Data integration: example



Create view object as

select * from v-object1
Union all
select * from v-object2

Create view source as

select * from v-source1
Union all
select * from v-source2

Data integration: example

Configuration	Mediator	data source 1	data source 2	Observation
Operating System	Ubuntu 12.04LST	Ubuntu 12.04LST	Ubuntu 12.04LST	...
Processor	4 VCPU ..GHZ	4 VCPU ..GHZ	4 VCPU ..GHZ	...
Memory	16 Go	16 Go	16 Go	...
Virtual HD	200 Go	200 Go	200 Go	...
Database	Oracle 11g.2	Oracle 11g.2	Oracle 11g.2	...
Dataset	–	PT1.1	PT1.1	Dataset of Petasky
Name of VM	fed	source1	source2	...
Schemas on DB	fed	chunk1	chunk2	...
Access tool	Sqldeveloper			...

Dataset	OBJECT table	SOURCE Table	Observation
Datafile size	4,5 Go	85 Go	...
Size in DB	2,5625 Go	60 Go	...
Row size in DB	686 bytes	391 bytes	...
Datafile name	Object.csv	Source.csv	...
Columns	227	92	...
Rows	4 012 341	164 989 583	...
Loading time with sqlldr	$\simeq$ 2h 30 mn	$\simeq$ 47h	using sqlloader
Copying from Oracle to Oracle database	under 10 mn	under 3h	using create table with database link

Select o.*, s.taimidpoint, s.psfflux from object o , source s
where s.objectid = o.objectid and ra_ps between 0.5 and 3
and decl_ps between -0.5 and 4 ;

31

Generated plan

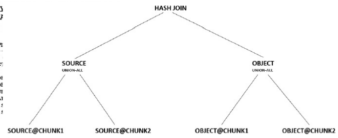
```
select o.*, s.taimidpoint, s.psfflux from source s, object o
where s.objectid = o.objectid
and ra_ps betwe
and decl_ps bet

call     count                                              urrent       rows
------- ------                                             ---------- ----------
Parse        1                                                      1          0
Execute      1                                                      0          0
Fetch  2049688                                                      0   30745295
------- ------  -------- ---------- ---------- ---------- ---------- ----------
total  2049690   2368.84    2903.63     691515          0          1   30745295

Misses in library
Optimizer mode:
Parsing user id:

Rows      Row Source Op
-------   ------------
30745295  HASH JOIN
  682656  VIEW  OBJECT
  682656   UNION-ALL
  341339    REMOTE  OB
  341317    REMOTE  OB
153123469  VIEW  SOUR
153123469   UNION-ALL
76561579    REMOTE  S
76561890    REMOTE  S
```

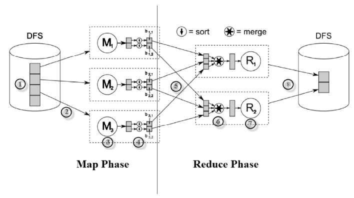
(+) Declarative mapping
(+) Query rewriting to generate plans
(-) Execution plans are far from being optimal



32

Mapreduce approaches

a new programming model intended to facilitate the development of scalable parallel computations on large server clusters



Mapreduce approaches

	Hadoop [Dean et al, 04]	Hadoop++ [Dittrich et al, 10]	HAIL [Dittrich et al, 12]	HadoopDB [Abouzeid et al, 09]	Hive [Thusoo et al, 10]
Query language	procedural	Procedural	Procedural	Declarative	Declarative
Simple indexes	Not supported	supported	Supported	supported	Supported
Multi indexes	Not supported	Not supported	supported	Supported	Supported
Complex indexes	Not supported	Not supported	Not supported	supported	Supported
Storage System	HDFS	HDFS	HAIL	Classical DBMS	HDFS
Index choice	-	Manual	Manual	Automatic	Manual

Description of the platform

The platform is formed of 10 DELL C6220 machines

Hardware configuration of machines

- 8 nodes of:
  - 24 processors (Xeon E5-2620 with 6 cores)
  - 6 disks of 1TB SATA 7200rpm
  - 3 nodes with 64 GB of RAM
  - 3 nodes with 92 GB of RAM
  - 2 nodes with128 GB of RAM
- 2 nodes of:
  - 24 processors (Xeon E5-2620 with 6 cores)
  - 2 disks of 1TB SATA 7200rpm
  - 1 node with 240 GB of RAM
  - 1 node with 32 GB of RAM (controller)
- Network Transfert rate: 2 Gb/s
- Virtualisation: OpenStack (Libvirt, Cepth, KVM, …)
- OS: Ubuntu 12.04

Resource	Available
RAM	1 TB
Disk	52 TB
#Processors	240

Data sets

- PT 1.1 and PT 1.2 (90 GB and 145 GB)
  - Mainly two tables: Source and Object
- Winter13: 3 TB
  - One Table => no join queries
- Data generation
  - 118 GB for the table Source (after elimination of sources without "ObjectId")
  - 7 GB for the table Object
  - increment of ID for the two tables
  - Same « RA »
  - +/- 9° for « DECL »
- Different sizes: 250 GB, 500 GB, 1 TB et 2 TB

Properties of generated data

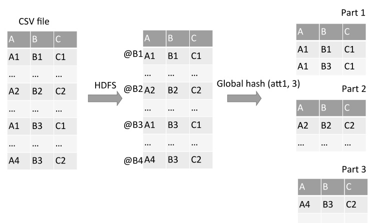
Query Id	Source: #records	Source:	Object:	Object:
PT12 250 GB	325196890	236 GB	9060260	14 GB
PT12 500 GB	650393780	472 GB	18120520	28 GB
PT12 1 TB	1300787560	944 GB	36241040	56 GB
PT12 2 TB	2601575120	1888 GB	72482080	112 GB

- RefSrcMatch: 1.7 GB pour 189 million records
- Science_Ccd_Exposure_Metadata: 16 GB pour 41 million records

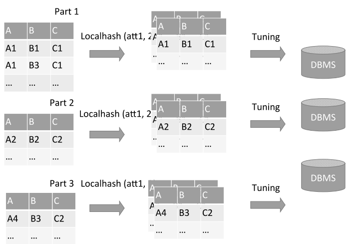
Data management tools

- Distributed systems
  - Map/Reduce: Hadoop
  - Hive: version 0.11 with Hadoop 1.1
  - HadoopDB: Hadoop 0.19
- Why Hive and HadoopDB ?
  - Support of declarative queries
  - Available RAM << Size of data on disk

Data load with HadoopDB



Data load with HadoopDB



Configurations

- Virtual machines
  - Clusters of 3, 6, 12, 25, 50 and 100 nodes
  - For each node: 8GB of RAM and 2 VCPU
  - Network transfert rate: 100 Mb/s
  - Disks: 350Go, 200Go et 100 Go
  - Ubuntu 12.04
- Hadoop:
  - HDFS: 256 MB
  - JVM memory: 1024 MB
  - Two instances and one instance Map Reduce are allowed to run simultaneously on each node.
  - Buffer for Postgresql (HadoopDB): 1 GB

Query: Syntax and Semantics (1/2)

	id	Syntaxe SQL
Selection	Q1	select * from source where sourceid=29785473054213321;
	Q2	select sourceid, ra,decl from source where objectid=402386896042823;
	Q3	select sourceid, objectid from source where ra > 359.959 and ra < 359.96 and decl < 2.05 and decl > 2;
Group By	Q4	select sourceid, ra,decl from source where scienceccdexposureid=454490250461;
	Q5	select objectid,count(sourceid) from source where ra > 359.959 and ra < 359.96 and decl < 2.05 and decl > 2 group by objectid;
Join	Q6	select objectid,count(sourceid) from source group by objectid;
	Q7	select * from source join object on (source.objectid=object.objectid) where ra > 359.959 and ra < 359.96 and decl < 2.05 and decl > 2;
	Q8	select * from source join object on (source.objectid=object.objectid) where ra > 359.959 and ra < 359.96;
	Q9	SELECT s.psfFlux, s.psfFluxSigma, sce.exposureType FROM Source s JOIN RefSrcMatch rsm ON (s.sourceId = rsm.sourceId) JOIN

Query: Syntax and Semantics (2/2)

	id	Syntaxe SQL
Order by	Q10	select objectid,sourceid from source where ra > 359.959 and ra < 359.96 and decl < 2.05 and decl > 2 order by objectid;
	Q11	select objectid,sourceid from source where ra > 359.959 and ra < 359.96 order by objectid;
UDF	Q12	select id FROM rundeepforcedsource where areaspec_box(coord_ra,coord_decl,-55,-2,55,2)=1;
	Q13	select fluxToAbMag(flux_naive) from rundeepforcedsource where objectid=1398583353936135;

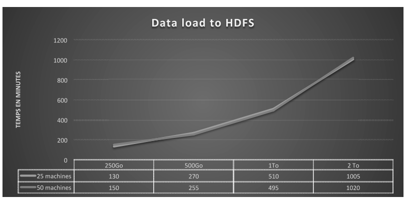
Query *v.s.* Data

Query Id	PT12 250 GB	PT12 500 GB	PT12 1 TB	PT12 2 TB
Q1	1	1	1	1
Q2	43	43	43	43
Q3	43	43	43	129
Q4	3650	7300	14600	29200
Q5	2	2	4	6
Q6	9060260	18120520	36241040	58892326
Q7	43	43	86	215
Q8	28576	28576	57152	127736
Q9	7763	7763	7763	7763
Q10	43	43	86	86
Q11	28576	28576	57152	114304

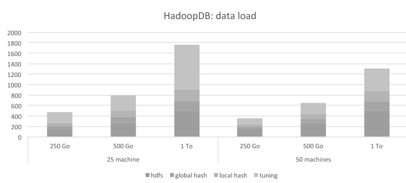
Index: some statistics

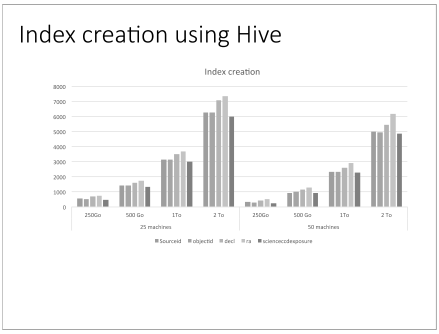
Index	PT12 250 GB	PT12 500 GB	PT12 1 TO	PT12 2 TB	relevant queries
SourceId	325196890	650393780	1300787560	2601575120	Q1
ObjectId	9060260	18120520	36241040	72482080	Q2,Q5,Q6,Q7, Q8,Q10 et
scienceccdexposureid	84785	84785	84785	84785	Q4
RA	162598427	162598427	162598427	162598427	Q3, Q5, Q7, Q8, Q9, Q10
DECL	325196854	650393708	1300787416	1300787416	Q3, Q5, Q7, Q9 et Q10

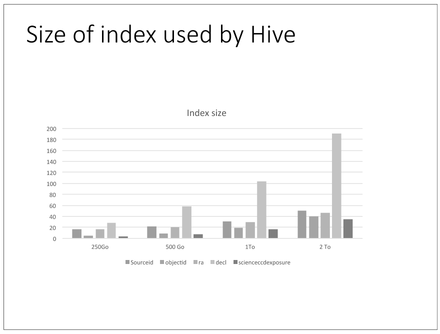
Data load with Hive

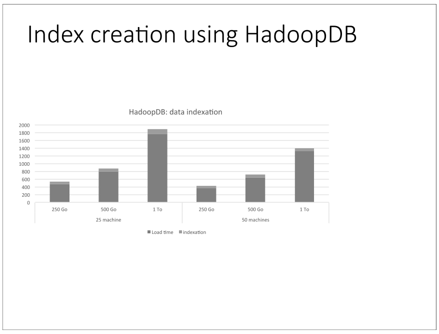


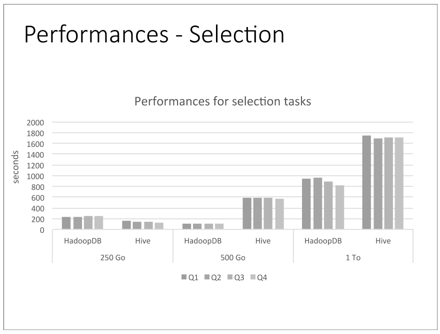
Data load with HadoopDB

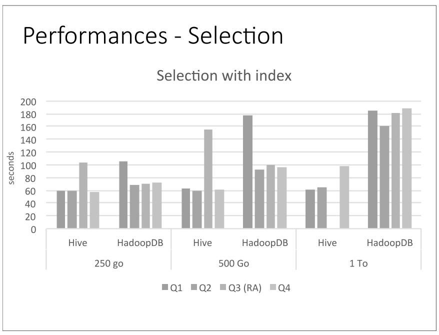


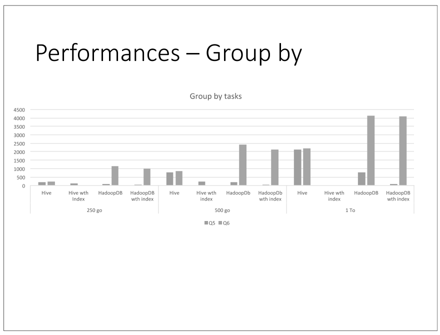

Index creation using Hive
Index creation
250Go
500 Go
1To
2 To
25 machines
50 machines
Sourceid
objectid
decl
ra
scienceccdexposure

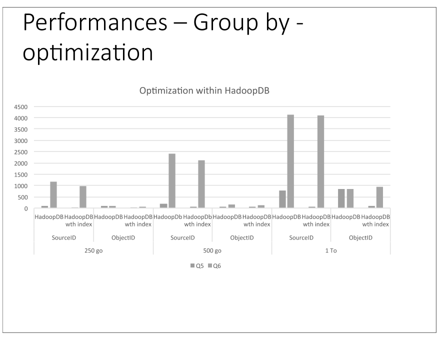

Size of index used by Hive
Index size
250Go
500 Go
1To
2 To
Sourceid
objectid
ra
decl
scienceccdexposure

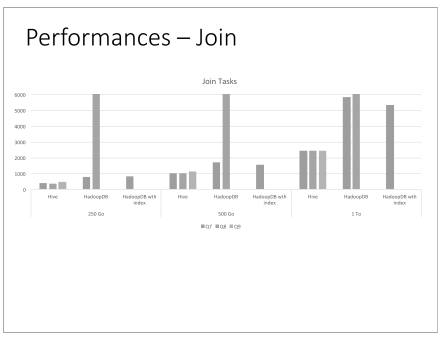

Index creation using HadoopDB
HadoopDB: data indexation
250 Go
500 Go
1 To
25 machine
50 machines
Load time
indexation


Performances - Selection
Performances for selection tasks
seconds
HadoopDB
Hive
250 Go
500 Go
1 To
Q1
Q2
Q3
Q4

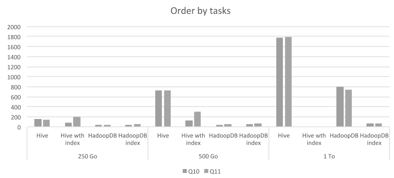

Performances - Selection
Selection with index
seconds
Hive
HadoopDB
250 go
500 Go
1 To
Q1
Q2
Q3 (RA)
Q4


Performances – Group by
Group by tasks
250 go
500 go
1 To
Q5
Q6

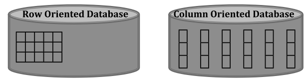

Performances – Group by - optimization
Optimization within HadoopDB
SourceID
ObjectID
250 go
500 go
1 To
Q5
Q6


Performances – Join
Join Tasks
Hive
HadoopDB
HadoopDB with index
250 Go
500 Go
1 To
Q7
Q8
Q9

Performances – Order by



Physical storage
Row store vs. column store



- High cost of I/O
- Analytical queries are expensive
- Problem with null values

- Cost of join

Experiments

- RowDB
  - ✓ Memory 8G
  - ✓ Processors: 1
- Column DB
  - ✓ Memory 8G,
  - ✓ Processors: 1
- Mediator :
  - ✓ Memory 16 G
  - ✓ Processors: 1

- Source : 70 GB , 92 columns , 165,000,000 Tuples
- Object : 5GB, 230 columns , 4,012,341 Tuples

Id	Query Statement
01	SELECT * FROM source WHERE sourceId = 2875056747578937;
02	SELECT sourceId, taimidpoint FROM source WHERE sourceId =386942193644115 AND SCIENCECCDEXPOSUREID=43856065114;
03	SELECT sourceId, taimidpoint FROM source WHERE sourceId =386942193644115;
04	SELECT taiMidpoint, modelFlux, modelFluxerr FROM source WHERE sourceId =386942193646211;
05	SELECT * FROM object WHERE ra_ps >= 1.0 AND ra_ps < 2.0 AND decl_sg > 1.0 AND decl_sg < 2.0;
06	SELECT * FROM source WHERE ra> 2.0
07	SELECT * FROM object WHERE rflags <40 AND iflags < 7000 AND yflags > 40000 AND zflags > 40000
08	SELECT objectId, count(sourceId) FROM source GROUP BY objectId;
09	SELECT * FROM source JOIN object on (source.objectId = object.objectId) WHERE source.objectId =386942193646211;
10	SELECT objectId, sourceId FROM source ORDER BY objectId;

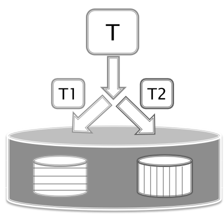
Execution Time

	Oracle	MonetDB
Q1	00:08:52.60	00:00:05.00
Q2	00:10:09.68	00:00:07.94
Q3	00:13:26.09	00:00:01.04
Q4	00:12:47.09	00:00:00.94
Q5	00:00:34.65	00:00:37.56
Q6	01:38:36.98	- (connection terminated)
Q7	00:01:03.40	00:00:07.50
Q8	00:15:11.16	01:20:00.00
Q9	00:12:56.47	00:00:49,00
Q10	01:11:10.63	00:03:55,00

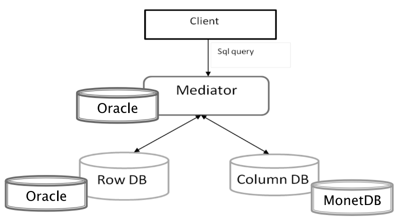
Execution Time: Latency

Column1	ORACLE	MonetDB
Q1	00:05:00,00	00:00:46.00
Q2	00:04:42,00	00:00:00.12
Q3	00:04:15,00	00:00:00.03
Q4	00:04:04,00	00:00:00.90
Q5	00:00:02,19	00:00:42.00
Q6	00:00:02,50	00:00:03,00
Q7	00:00:18,14	00:00:6.3
Q8	00:07:38,00	00:00:39.2
Q9	00:01:39,00	00:00:45.00
Q10	00:04:03,00	00:08:13,00

Hybrid storage



Implementation of a hybrid system using an information integration approach



Id	Query Statement
01	SELECT * FROM source WHERE sourceId = 2875056747578937;
02	SELECT sourceId, taimidpoint FROM source WHERE sourceId =386942193644115 AND SCIENCECCDEXPOSUREID=43856065114;
03	SELECT sourceId, taimidpoint FROM source WHERE sourceId =386942193644115;
04	SELECT taiMidpoint, modelFlux, modelFluxerr FROM source WHERE sourceId =386942193646211;
05	SELECT * FROM object WHERE ra_ps >= 1.0 AND ra_ps < 2.0 AND decl_sg > 1.0 AND decl_sg < 2.0;
06	SELECT * FROM source WHERE ra> 2.0
07	SELECT * FROM object WHERE rflags <40 AND iflags < 7000 AND yflags > 40000 AND zflags > 40000
08	SELECT objectId, count(sourceId) FROM source GROUP BY objectId;
09	SELECT * FROM source JOIN object on (source.objectId = object.objectId) WHERE source.objectId =386942193646211;
10	SELECT objectId, sourceId FROM source ORDER BY objectId;

Execution Time

	Oracle	MonetDB	Hybrid
Q1	00:08:52.60	00:00:05.00	00:09:00.00
Q2	00:10:09.68	00:00:07.94	00:03:15.00
Q3	00:13:26.09	00:00:01.04	00:00:00.76
Q4	00:12:47.09	00:00:00.94	00:03:31.00
Q5	00:00:34.65	00:00:37.56	00:34:00.00
Q6	01:38:36.98	- (connection terminated)	-
Q7	00:01:03.40	00:00:07.50	00:00:16.56
Q8	00:15:11.16	01:20:00.00	02:30:00.00
Q9	00:12:56.47	00:00:49,00	00:11:08.00
Q10	01:11:10.63	00:03:55,00	01:44:25.00

Execution Time: Latency

Column1	ORACLE	MonetDB	Hybrid
Q1	00:05:00,00	00:00:46.00	00:06:00,00
Q2	00:04:42,00	00:00:00.12	00:02:28,00
Q3	00:04:15,00	00:00:00.03	00:00:01,00
Q4	00:04:04,00	00:00:00.90	00:03:40,00
Q5	00:00:02,19	00:00:42.00	00:05:06,00
Q6	00:00:02,50	00:00:03,00	00:39:00,00
Q7	00:00:18,14	00:00:6.3	00:00:18,00
Q8	00:07:38,00	00:00:39.2	01:40:00,00
Q9	00:01:39,00	00:00:45.00	
Q10	00:04:03,00	00:08:13,00	00:00:09,00

Learned lessons and research directions

No one fits all

- MapReduce-based algorithms can be useful to implement physical operators
- Hybrid system: row/column store
- Need for more research on
  - ✓ Abstraction adequate to the scientific domain
  - ✓ Support of user defined functions
  - ✓ Optimization techniques embedded in the data management system
  - ✓ Scalability of information integration framework

Existants

- Druide : a scan rate of 26 billions records per second, with our distributed, in-memory data store called Druid.
  - ➡ cluster of 100 nodes, each with 16 cores, 60GB of RAM, 10 GigE ethernet, and 1TB of disk space
  - ➡ Collectively the cluster comprised 1600 cores, 6TB of RAM, fast ethernet and more than enough disk space.
- Google Dremel
- Oracle RAC

Data management challenges in LSST

Complex computations over complex data

- Queries over hundreds of attributes
- Real time analysis of 2 TB/hour
- Real time monitoring of tens of billions of objects
- Some typical queries
  - Point-query (looking for a needle in a haystack)
  - Correlations : join queries over **10^9 galaxies**
  - Time serie : 10 years of data, 1000 visits par pointé, coaddition d'images, soustraction d'images, ...