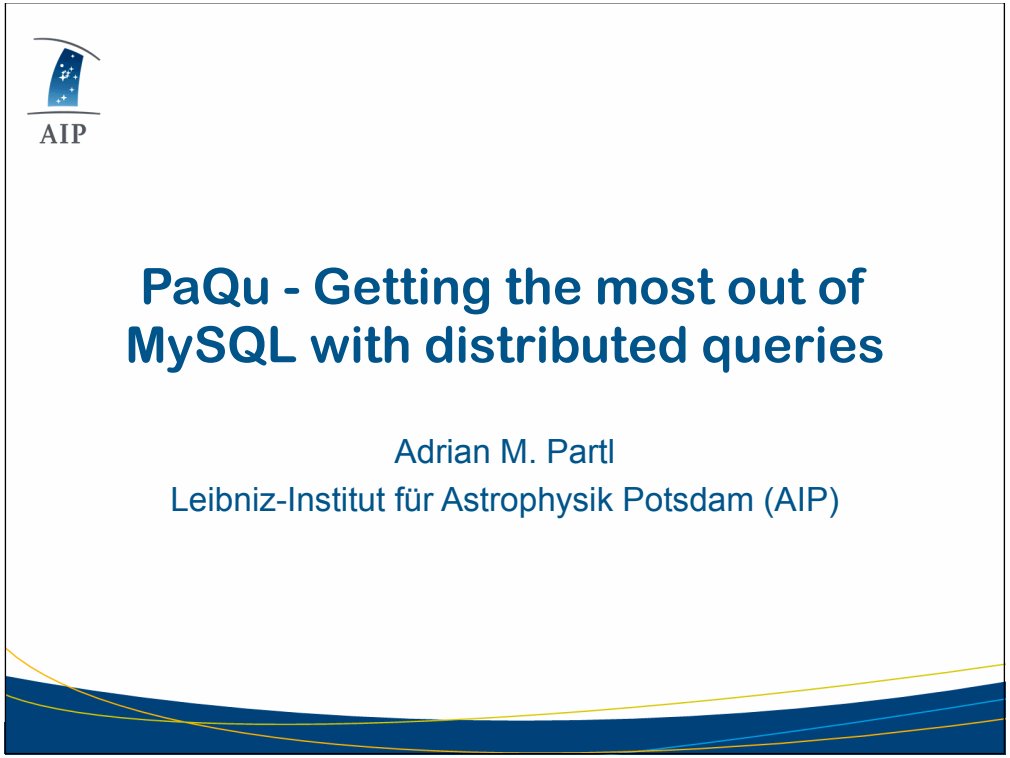
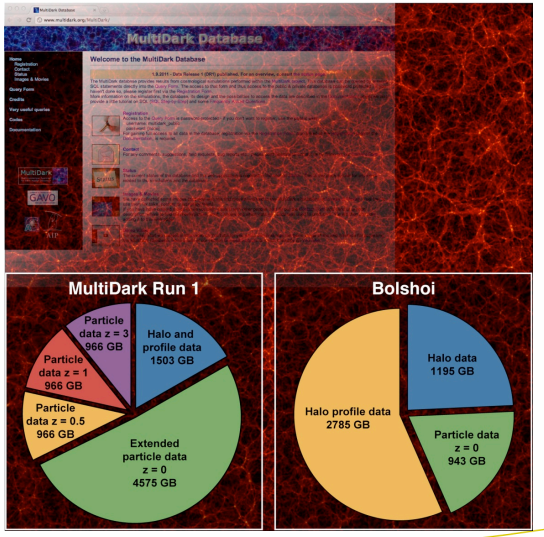

AIP
PaQu - Getting the most out of MySQL with distributed queries
Adrian M. Partl
Leibniz-Institut für Astrophysik Potsdam (AIP)

www.multidark.org

- Total row count: $2.34\ 10^{11}$
- Most queries 100 sec
- significant amount of queries > 1000 sec
- Full table scan for largest tables: ~ 30 minutes



Problems while building the MultiDark DB

- Data ingest time:
  Need to convert binary to ASCII CSV format
  (highly inefficient)
- Data transformation:
  Computing values after ingest slow - best during ingest
- Data indexing:
  Index on particle data (~10^{10} particles) around one week
- Data retrieval times slow on full table scans (~30 min):
  cannot build index for every query
- Spatial queries in 3D hard, impossible in 6D
  nearest neighbour search also inefficient

Why RDBMS?

- SQL - it took long time for the community to adopt SQL (we think this is the main problem with NoSQL)
- proven, widely available, large user base
- good for structured data

- Problems:
  - Built for different purposes (business, web, ...) result sets usually small - mostly in memory solutions
  - parallelisation of data / sharding
  - can be expensive

Our vision:

- Open source DB solution for scientific purposes:
  A one size fits all solution built by the community for the community

- Developments at AIP:
  - DB independent ingestion library and data transformation tool *(DBIngestor and AsciiIngest)*
  - MySQL job queue *(mysql_query_queue)*
  - MySQL sharding solution for scientific queries *(PaQu)*
  - Future*:*
    - MySQL plugins for data analysis, spatial queries and indexing
    - MySQL storage engine plugins for simulation raw data

PaQu - HPQ for MySQL

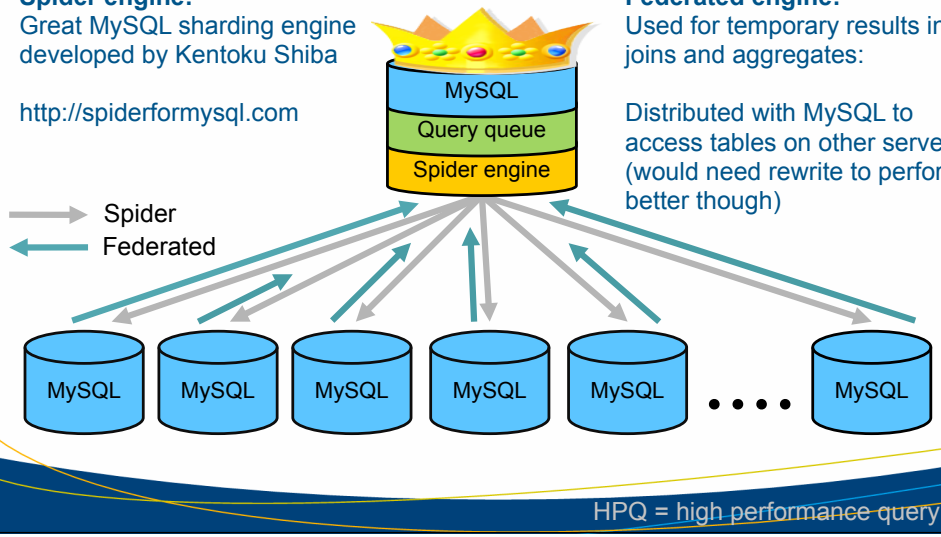
- Full table scan of large table: ~ 30 minutes
- From HPC we know:
  Distribute your problem over N nodes and you are N times faster (if you are lucky)
  – i.e. on 10 nodes, full table scan of large table:
    3 minutes!
- From Big Data hype we know:
  Use Hadoop and everything is fine
- We want SQL! We want “Hadoop” for MySQL!

HPQ = high performance querying

PaQu - HPQ for MySQL

Spider engine:
Great MySQL sharding engine developed by Kentoku Shiba

http://spiderformysql.com

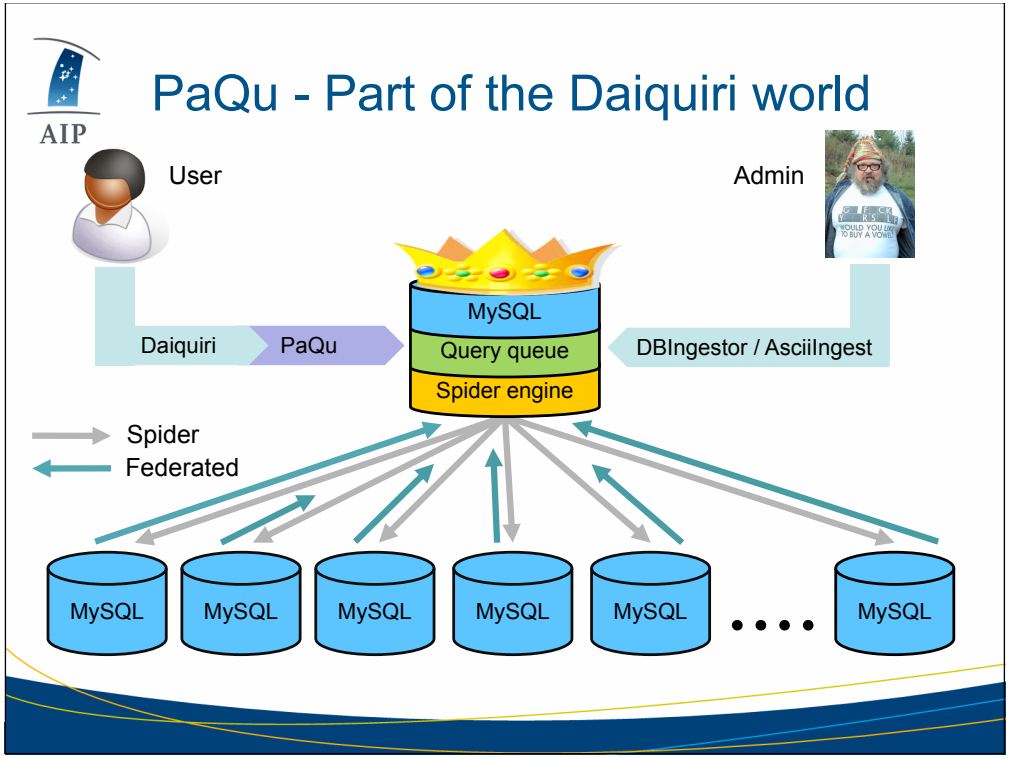


Federated engine:
Used for temporary results in joins and aggregates:

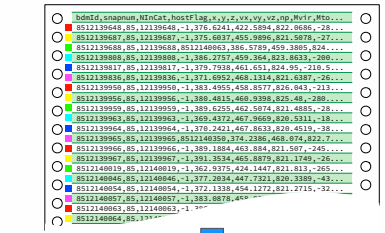
Distributed with MySQL to access tables on other servers. (would need rewrite to perform better though)

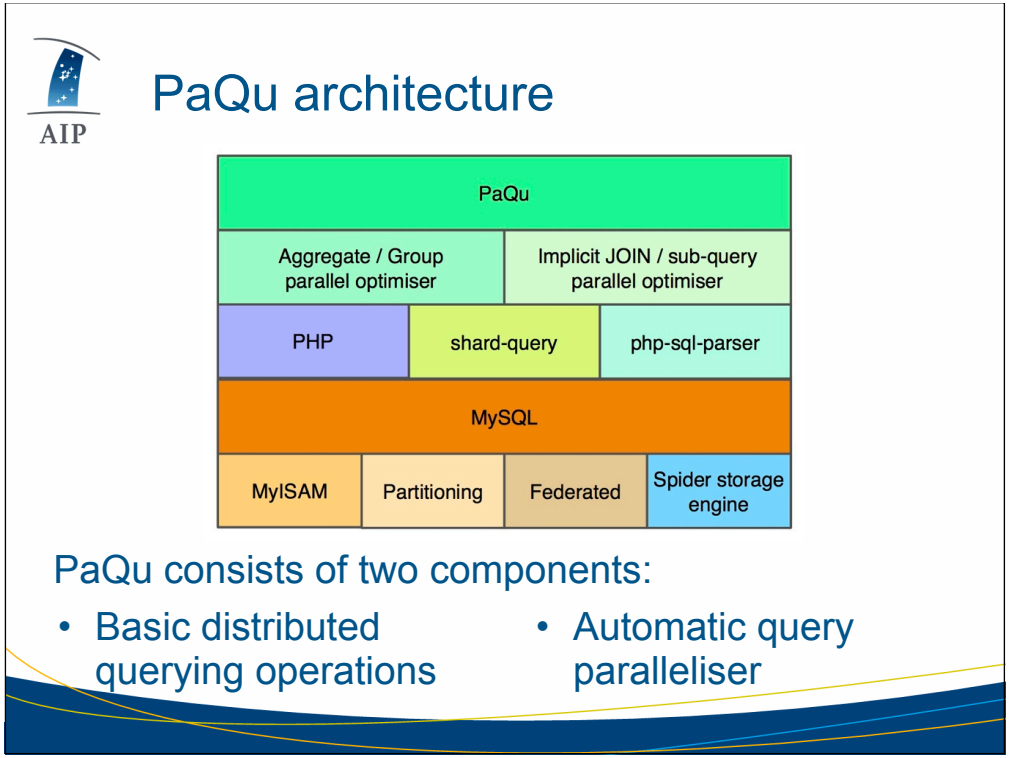
HPQ = high performance querying

PaQu - Part of the Daiquiri world



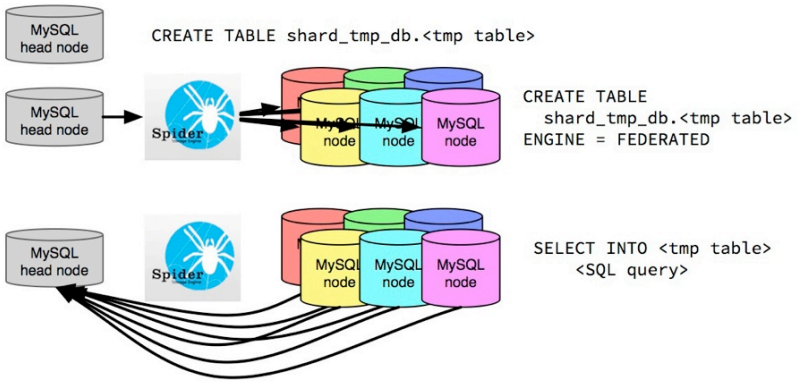
Sharding


bdmId,snapnum,NInCat,hostFlag,x,y,z,vx,vy,vz,np,Mvir,Rto...
AsciiIngestor
DBIngestor
MySQL
Spider engine
Partitioning
Spider
MySQL node
MySQL node
MySQL node
MySQL node
MySQL node
MySQL node


AIP
PaQu architecture
PaQu
Aggregate / Group parallel optimiser
Implicit JOIN / sub-query parallel optimiser
PHP
shard-query
php-sql-parser
MySQL
MyISAM
Partitioning
Federated
Spider storage engine
PaQu consists of two components:
• Basic distributed querying operations
• Automatic query paralleliser

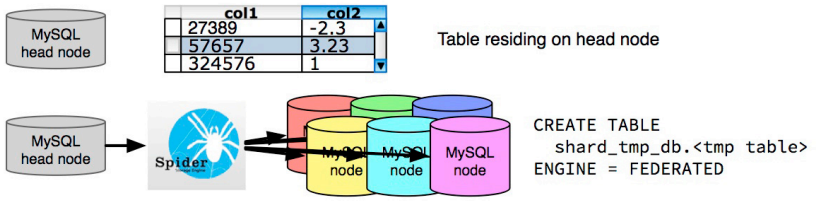
Basic distributed operations

CALL paquExec(<SQL query>, <tmp table>)



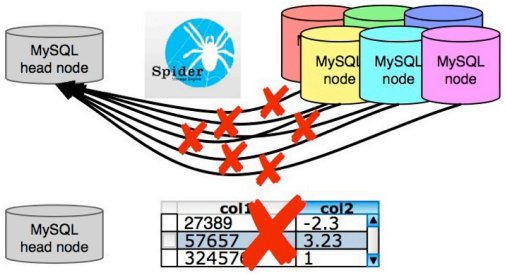
Basic distributed operations

```
CALL paquLinkTmp( <tmp table> )
```



Basic distributed operations

```
CALL paquDropTmp( <tmp table> )
```



DROP TABLE <tmp table>

DROP TABLE <tmp table>

Query paralleliser - example

```
SELECT   snapnum,
         AVG(mass)
FROM     MDR1.FOF
GROUP BY snapnum;
```

```
SELECT   a.snapnum, SUM(a.mass)/SUM(a.mass)
FROM
   (SELECT   snapnum, SUM(mass) as sum,
             COUNT(mass) as cnt
    FROM     MDR1.FOF
    GROUP BY snapnum) as a
GROUP BY  a.snapnum;
```

```
CALL paquExec(
           "SELECT snapnum AS `snapnum`, COUNT(mass) AS `cnt_avg(mass)`,
            SUM(mass) AS `sum_avg(mass)`
            FROM MDR1.FOF GROUP BY snapnum", "aggregation_tmp_8896713");

USE spider_tmp_shard; SET @i=0;
CREATE TABLE multidark_user_admin.`/*@GEN_RES_TABLE_HERE*/` ENGINE=MyISAM
     SELECT DISTINCT @i:=@i+1 AS `row_id`, `snapnum`,
                     (SUM(`sum_avg(mass)`) / SUM(`cnt_avg(mass)`)) AS `avg(mass)`
     FROM `aggregation_tmp_8896713` GROUP BY `snapnum` ;

CALL paquDropTmp("aggregation_tmp_8896713");
```

Automatic query paralleliser

Implicit Joins:

```
SELECT    a.*, b.*, c.*
FROM      a, b, c
WHERE     b=2 AND
  b.id=c.b_id AND
  a.id=b.a_id;
```

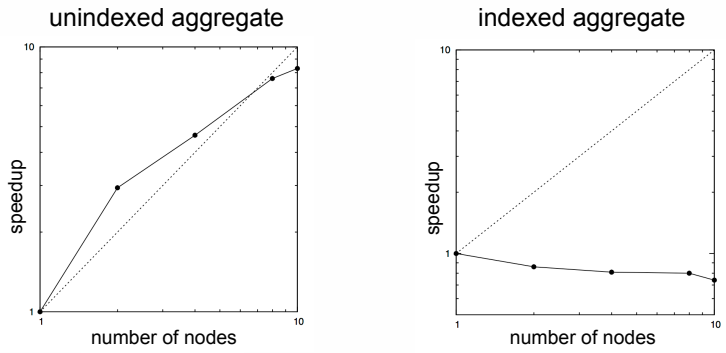
```
SELECT    a.*, tmp.*
FROM      a,
    (SELECT  b.*, c.*
     FROM    c,
             (SELECT  b.*
              FROM    b
              WHERE b=2) as b
     WHERE   b.id=c.b_id)
WHERE     a.id=tmp.b.a_ids
```

Aggregates:

```
SELECT    a.bar, AVG(a.foo)
FROM      a
GROUP BY  a.bar;
```

```
SELECT    a.bar,
          SUM(a.sum)/SUM(a.cnt)
FROM
   (SELECT   a.bar as bar,
             SUM(a.foo) as sum,
             COUNT(a.foo) as cnt
    FROM     a
    GROUP BY a.bar) as a
GROUP BY  a.bar;
```

Performance results

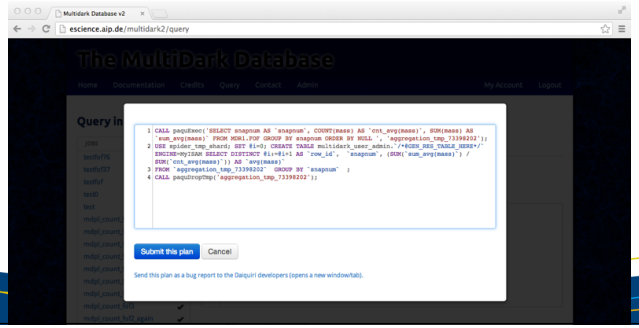


- Strong correlation with hardware setup:
  - Cache sizes, size of data files (smaller is better / partitioning?), network and I/O performance

MultiDark v2.0

- PaQu covers most known use cases of MultiDark
- All data from MultiDark now on 10 DB nodes
- PaQu fully integrated with Daiquiri

- http://escience.aip.de/multidark2



Conclusions

PaQu brings parallel querying to MySQL.

PaQu relies on open source.
PaQu is open source!

Download our DB developments here:
https://github.com/adrpar

Try PaQu yourself and compare with the original:
http://escience.aip.de/multidark2