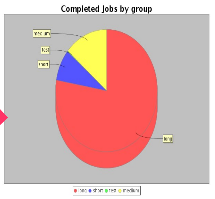
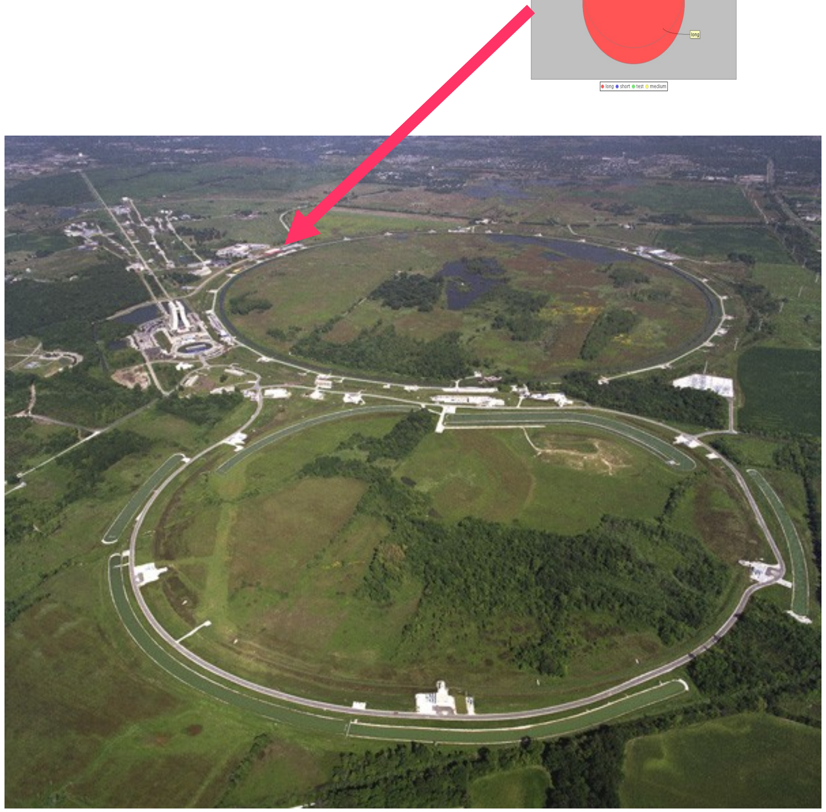
# Monitoring the CDF analysis farm (CAF)

CHEP 2009 21 - 27 March 2009 Prague, Czech Republic





**Marian Zvada**

**Hans Wenzel, Federica Moscato**

**Fermilab**

**March 26th, 2009**

# The CDF Central Analysis Farm (CAF)

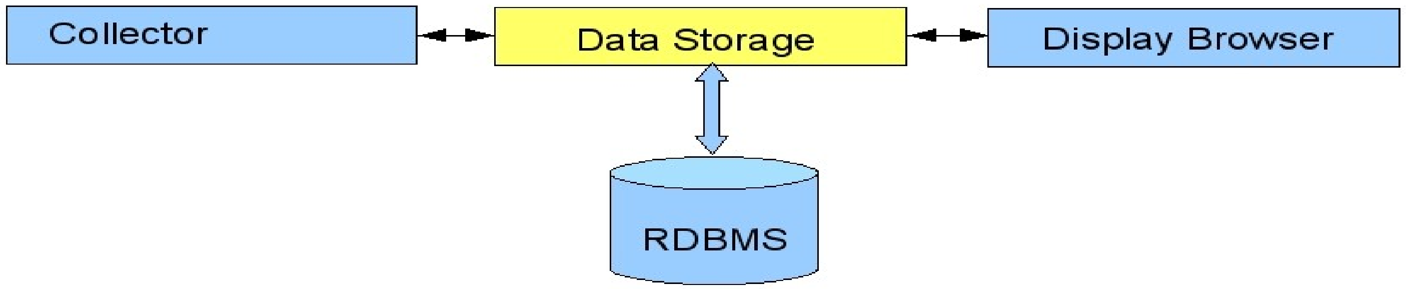
- CDF: active experiment at the Fermilab Tevatron, currently taking data producing more than 60 physics publications/year
- CAF supports 400 CDF physicists providing computing resources for: user data analysis, Monte Carlo production and Event reconstruction. (currently > 5000 slots)
- CAF is middle ware on the top of Condor batch system and CDF Grid infrastructure.
- CDF is in the process to move from dedicated computing resources to shared centrally managed computing resources.

## Stakeholders

- Users/Physicist: want to know how their jobs are doing.
  - Why do jobs fail? Application or System at fault?
  - Estimate resources needed for big productions
- Administrators: Want to know the Status of the farm
- Management: Want to know if the resources are utilized and if they have to provide more

Application developers should instrument the Programs with proper return codes making it possible to analyze the reason for the Application failure.

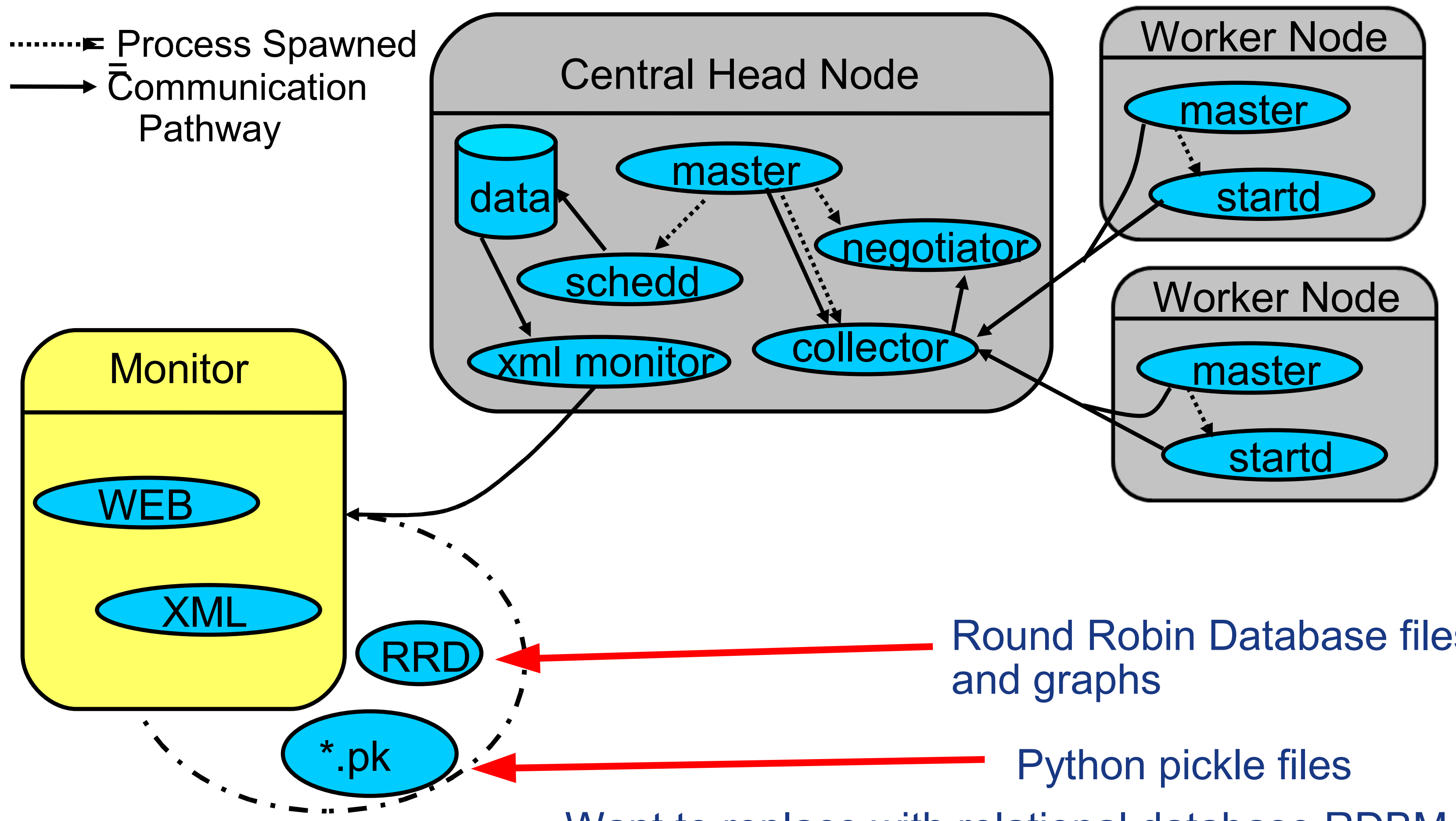
# Elements of a monitoring System



- **Collector:** are processes/agents that collect the monitoring data. The collectors must be robust and lightweight and should have very little impact on the processes they are monitoring.
- **Data storage:** way to store the monitoring data in the most convenient way
- **Display Browser:** this is the process making the data available on the web. There is a large number of Modern web application frameworks that use data bases as their back ends. The display and Data storage components should be clearly separated.

# Current CAF monitor framework

Provides static html web pages:



Round Robin Database files and graphs

Python pickle files

Want to replace with relational database RDBM

## Why RDBM?

- more robust
- the data source for the monitoring is consolidated and browse-able (SQL queries).
- allows for flexible queries and dynamic view in addition to the current static views.
- many web based tools and software frameworks are available to access and visualize db contents
- data from different monitoring sources can be cross referenced and compared. (Zabbix, Condor (Quill), CAF...)
- after the data is collected and stored in the data base the monitoring will cause no load on the system: all communication is done with the (separate) data base server.

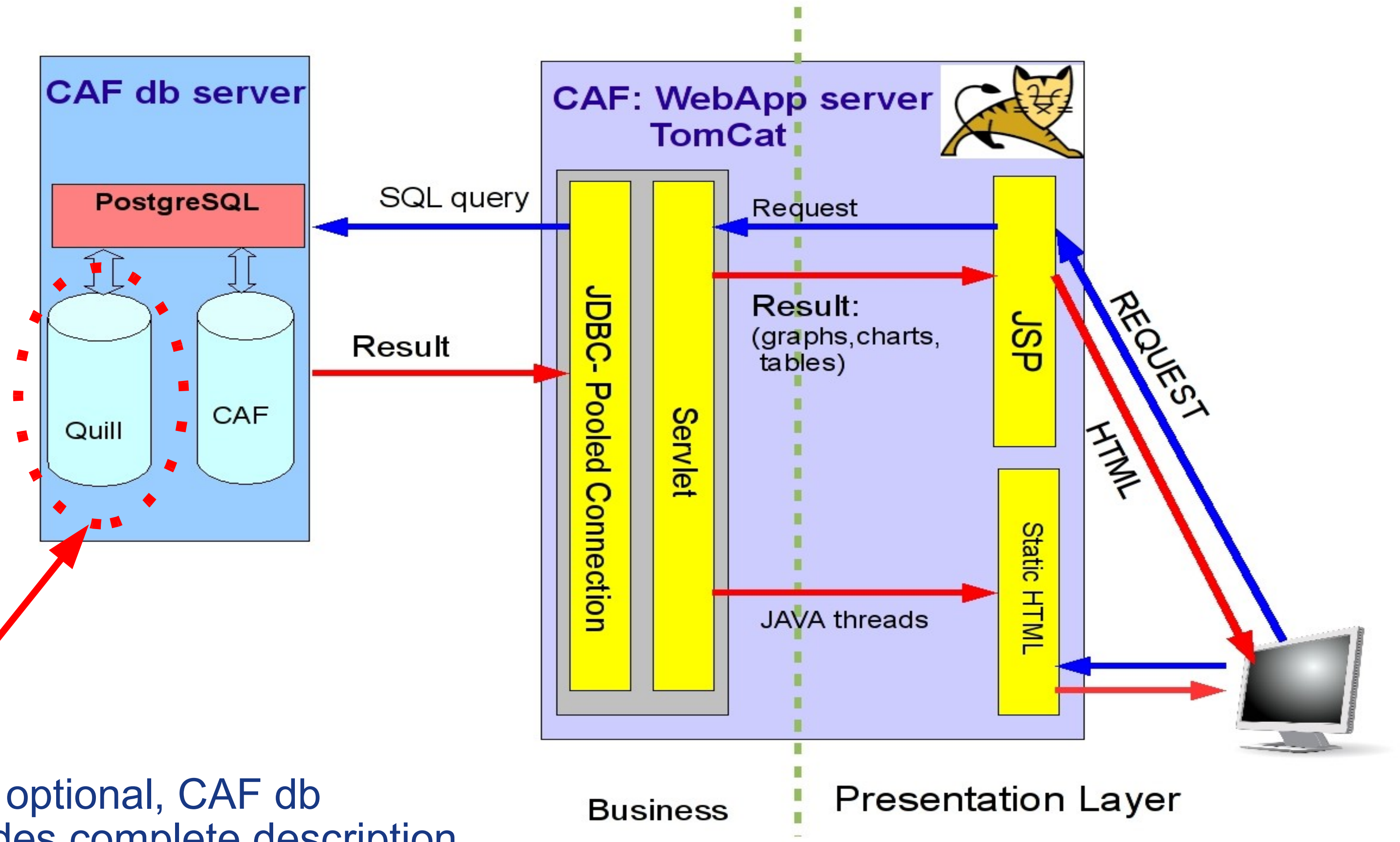
## Why RDBMs (cont.) ?

- Tools exist to backup the data, condense older data, synchronize databases  etc.
- Database engines provide the infrastructure to control who and from where the database can be accessed.
- Database maintains referential integrity of the data.
- allows to automate specific tasks like generating reports, re-indexing etc.

# Condor Quill for data collection?

- The Good (Il Buono):
  - provided by condor team → Condor specific
  - A non-invasive method of storing a read only version of the job queue and job historical data in a relational database (POSTGRES). Now quill++ stores entire condor information.
  - presents the job queue information as a set of tables in a relational database.
  - provides performance enhancements in very large and busy pools.
  - hook up to monitoring info when available → in general provides monitoring for non CAF specific condor pools.
- The Bad (Il Cattivo) and the Ugly (Il Brutto):
  - stores lots of data we are not interested in
  - some information we are interested in is missing
  - can't run collectors on worker nodes (not using dedicated farms) → not a complete solution,
  - doesn't apply to compute farms not based on condor.

## Architecture of new CAF Monitor



Quill: optional, CAF db provides complete description of monitoring info

# Tools and development environment

- Monitor is Java/JSP based web application
- Use:
  - Netbeans IDE: comes with all the tools to deploy and build, internal tomcat server etc. etc.
  - PostgreSQL database to store the data
  - Tomcat Web server
  - JDBC database access (pooled connection)
  - Use JSTL tag library for db access and table formatting. (less and less)
  - Access Java free chart via Java servlets for graphics with database as data input. Or create graphs with plain Java code.
  - Using Java threads and web application start up directives to automate processes.

# Some Data we collect

SQL schema for Job information:

```
CREATE TABLE JID
(
id                integer  NOT NULL,
dhmode            varchar(100),
group_name        varchar(10),
schedd_node       varchar(20),
prio              integer,
iomonfile         varchar(100),
errfile           varchar(100),
user_domain       varchar(100),
cmdline           varchar(300),
end_time          numeric(38),
schedd_name       varchar(20),
iomapfile         varchar(100),
caf_submit        numeric(38),
acctuser          varchar(100),
maxtime           numeric(38),
logfile           varchar(100),
email             varchar(100),
outurl            varchar(100),
user_name         varchar(100),
Primary Key (id)
);
```

Job can consist of many sections that might depend on each other:
Dagman

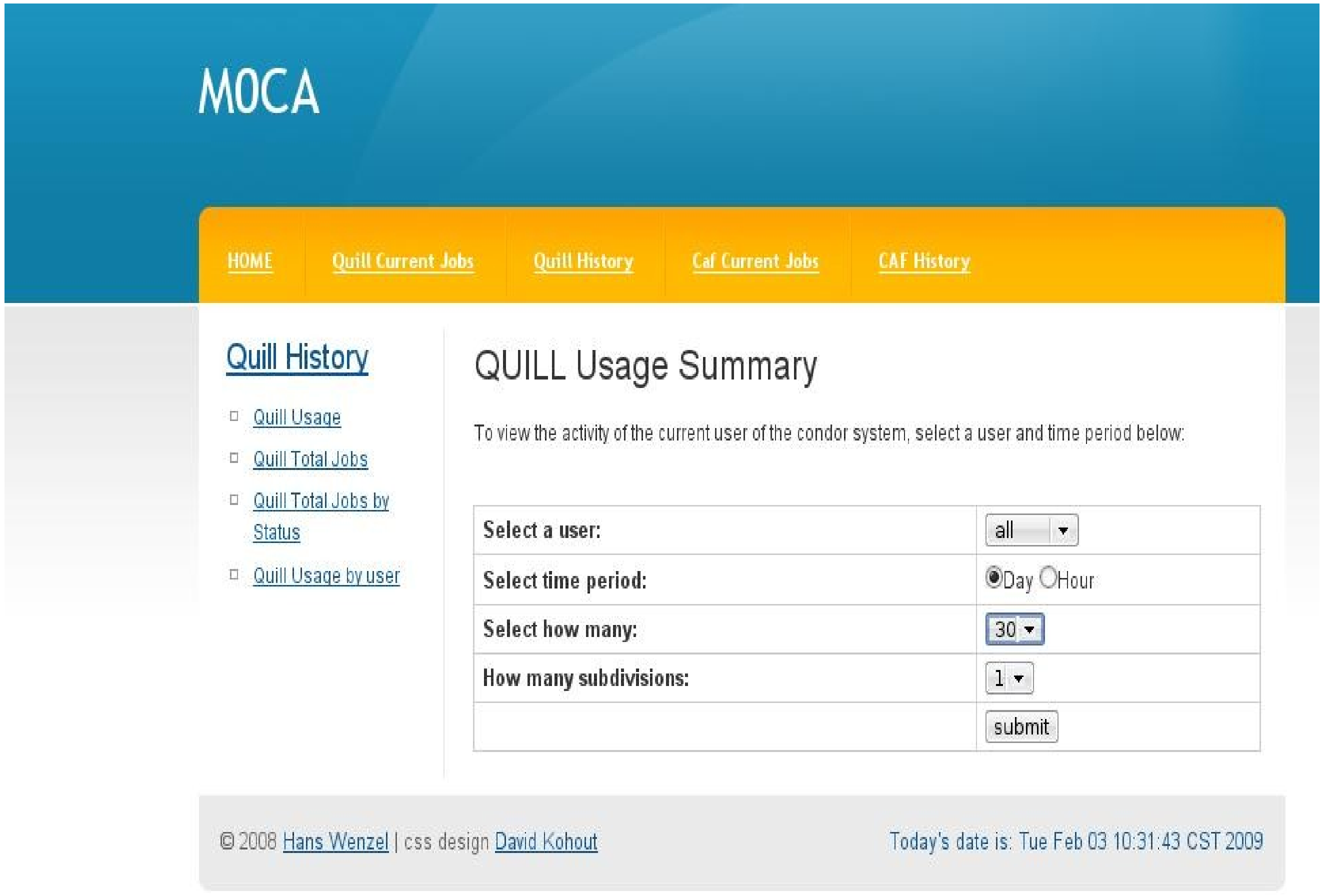
## Some data we collect (cont.)

SQL schema for Summary and section information:

```
CREATE TABLE SECTION_SUMMARY (
id                 integer  NOT NULL references JID(id),
total_main_utime   numeric(38),
total_utime        numeric(38),
max_mem            integer,
total_rtime        numeric(38),
datasets_str       varchar(100),
nr_sections        integer,
main_max_mem       integer,
main_exe           varchar(100),
Primary Key (id)
);
CREATE TABLE SECTIONS (
id                 integer  NOT NULL references JID(id),
secnr              integer  NOT NULL,
status             integer,
node               varchar(20),
nodename           varchar(20),
prio               integer,
start_time         numeric(38),
vm                 integer,
elapsed            numeric(38),
condor_id          integer,
assign_time        numeric(38),
nr_holds           integer,
nr_starts          integer,
lost_secs          integer,
submit_time        numeric(38),
nr_suspends        integer,
port               integer,
waited             numeric(38),
Primary Key (id,secnr)
);
```

Summary

Information about each section



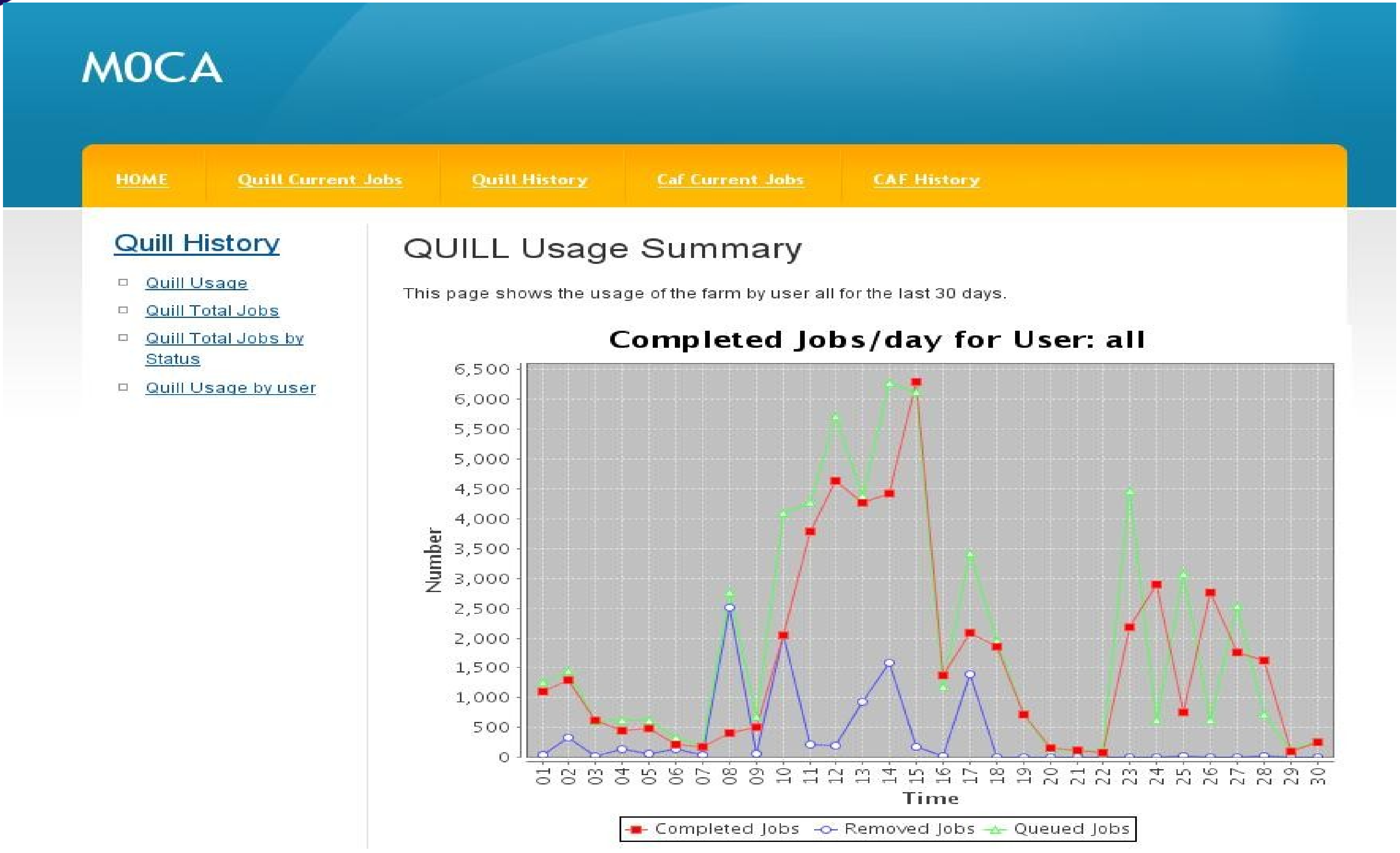
QUILL Usage Summary

To view the activity of the current user of the condor system, select a user and time period below:

| | |
|---|---|
| Select a user: | all |
| Select time period: | [✓ Day] Hour |
| Select how many: | 30 |
| How many subdivisions: | 1 |
| | submit |

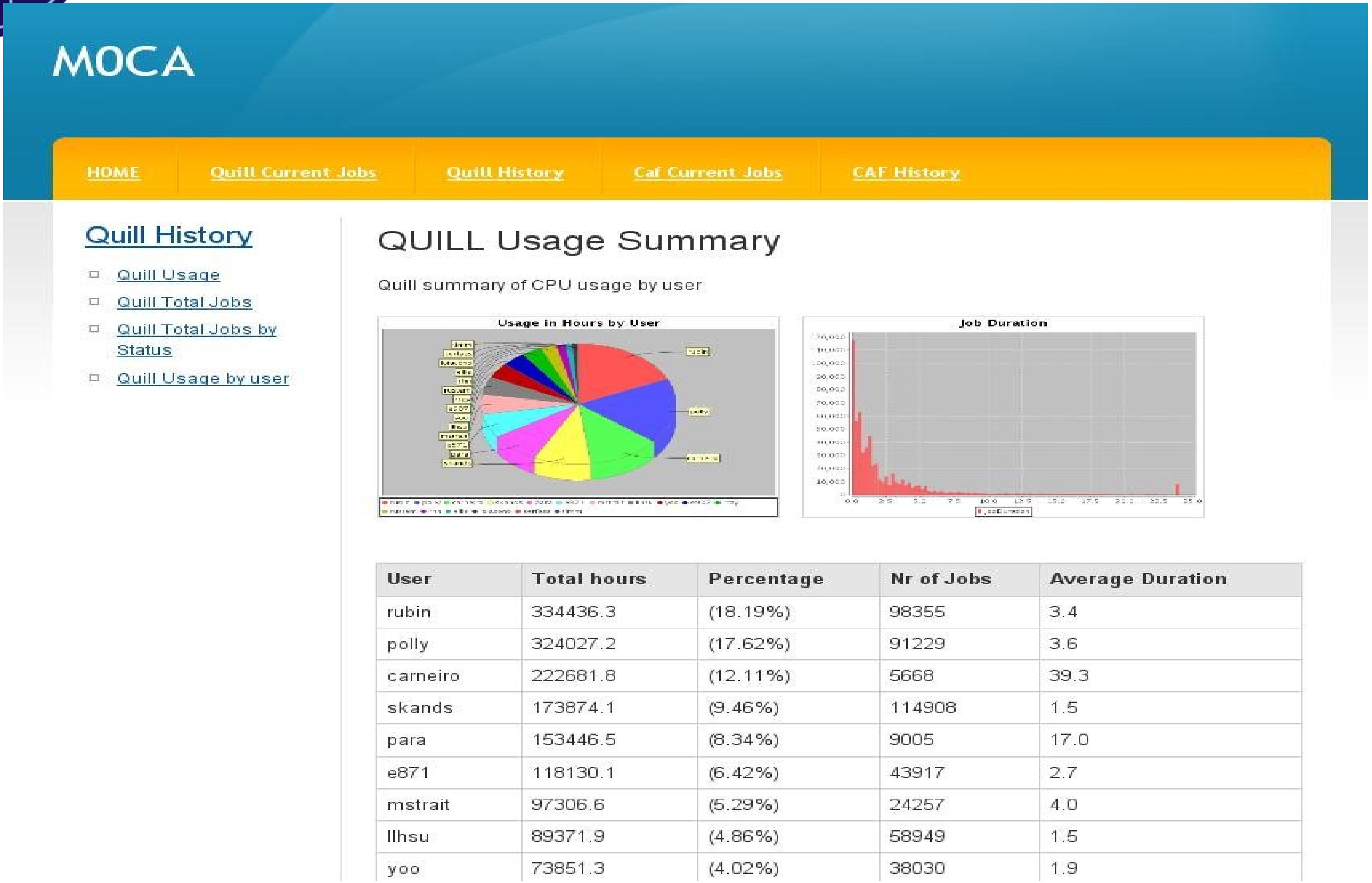
© 2008 Hans Wenzel | css design David Kohout

Today's date is: Tue Feb 03 10:31:43 CST 2009



## QUILL Usage Summary

This page shows the usage of the farm by user all for the last 30 days.



MOCA

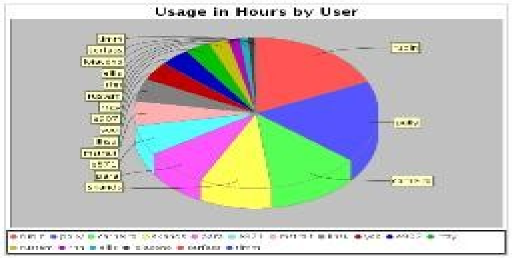
HOME | Quill Current Jobs | Quill History | Caf Current Jobs | CAF History

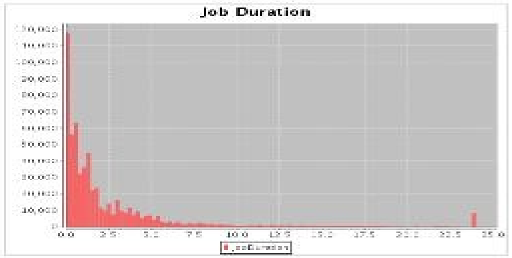
## Quill History

- Quill Usage
- Quill Total Jobs
- Quill Total Jobs by Status
- Quill Usage by user

## QUILL Usage Summary

Quill summary of CPU usage by user





| User | Total hours | Percentage | Nr of Jobs | Average Duration |
|---|---|---|---|---|
| rubin | 334436.3 | (18.19%) | 98355 | 3.4 |
| polly | 324027.2 | (17.62%) | 91229 | 3.6 |
| carneiro | 222681.8 | (12.11%) | 5668 | 39.3 |
| skands | 173874.1 | (9.46%) | 114908 | 1.5 |
| para | 153446.5 | (8.34%) | 9005 | 17.0 |
| e871 | 118130.1 | (6.42%) | 43917 | 2.7 |
| mstrait | 97306.6 | (5.29%) | 24257 | 4.0 |
| llhsu | 89371.9 | (4.86%) | 58949 | 1.5 |
| yoo | 73851.3 | (4.02%) | 38030 | 1.9 |

## Summary

- Developed Java based WebApplication with RDBM as backend to monitor the CAF.
- Work in progress:
  - Up to now only deployed on a small test farm
  - Still need to deploy on the production system
  - Test scaling.
  - Implement more views.