Using the LHC Computing Grid for the Measurement of the Top Quark Mass in the Dilepton Channel with the Matrix Element Method

Aurélien Croc

2 avril 2009

Outline

- Why do we need the grid?
- How can we use the LCH Computing Grid?
- The matrix element method
  - What is the method I'm working on?
  - The Moriond results for the dilepton channel
  - My first results
- Conclusion

Why Do we Need the Grid?

The Matrix Element Method

Issue:

The Matrix Element Method requires multi-dimensional integrations with Monte-Carlo method which is *very CPU consuming*.

The study needs to integrate (for $\mu\mu$ channel):

- Approximately **1000** Monte-Carlo events
- **5** simulated input top quark masses
- **15** tested masses values
- The background & the data
- The same amount of events for systematics (more than **5** systematics)

$\Longrightarrow$ More than **500 000** events to integrate (for **one** decay channel)

Time needed to integrate an event:

approximately **20** minutes

$\Longrightarrow$ More than **160 000 CPU hours** needed

*On CAB with 600 CPUs, it would take more than **11 days!***

The LCG Grid and How to Manage it

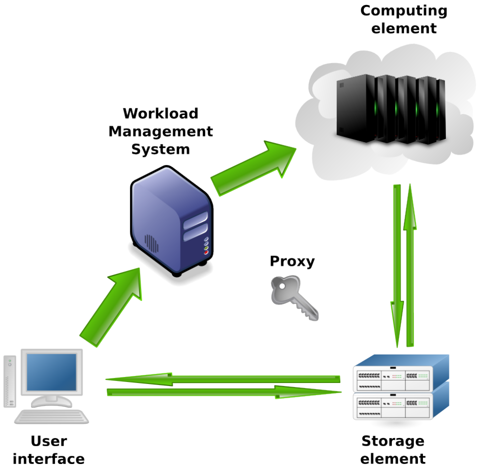
The Grid in Picture

A grid = multiple computing elements linked together by a network

A part of the LHC Computing Grid (LCG)

How to Submit a Job on LCG with the Official Tools

An overview of the *official* LCG job submission:



Job execution sequence:

- The user send a job to the workload management system (WMS)
- The WMS selects a remote computing element and send it the job
- The remote computing element executes the job
- The user fetchs the result

All the job management is done through the *GLite* tools

Submit a Job on Grid through a Workload Management System

The aim of the Workload Management System (WMS) is to send the submitted jobs to the best remote computing elements (environment, resources) in order to be executed.

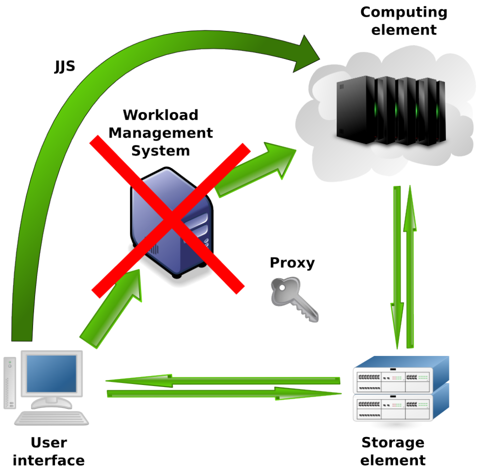
Common issues:

- From our experience, the choice of the remote computing element where the job will be sent isn't the best (jobs are always sent on the same remote computing elements whereas other with free CPU are available but are not used for unknown reasons)
- There is no way to know the status of each queue for each remote computing element in real time
- The workload management system can be easily overloaded when several thousand of jobs have to be managed

Solution:

Don't use the workload management system! We are using another job management system which will be shown in the next slides.

An overview of the Java Job Submission[1] (JJS) mechanism:



Job execution sequence:

- The user sends jobs to the JJS job manager
- JJS uses jobs as *probes* to know the current state of the remote computing elements
- If the job is in waiting queue since too long time, it is automatically moved and the remote computing element state is changed as *busy*
- Once the job is executed the user fetches the result

[1] Developed by Pascal Calvat, CNRS, France

My Work on JJS and its future for the DØ collaboration

JJS has not been developed for long-time jobs. Thus I've done modifications for that. It is now *almost ready* to use.

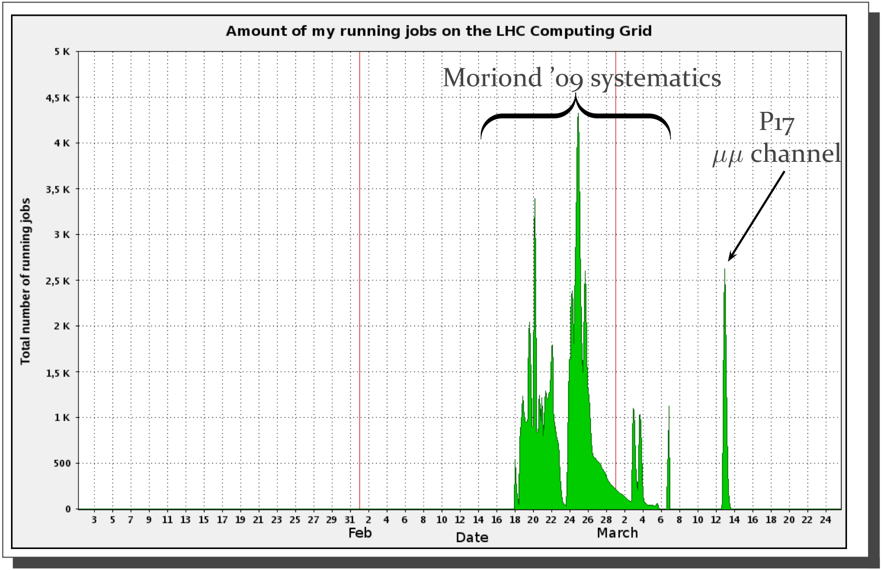
A non-exhaustive list of my contributions:

- Jobs submission algorithm improved
- Job resubmission algorithm improved
- File transfer improved (to prevent from overloaded storage element)
- Monitoring interface developed
- Debugging of the LCG to OSG bridge to submit job on the Open Science Grid (still working on it)
- bug solved

Outlook:

I am working to provide it to the DØ collaboration very soon.

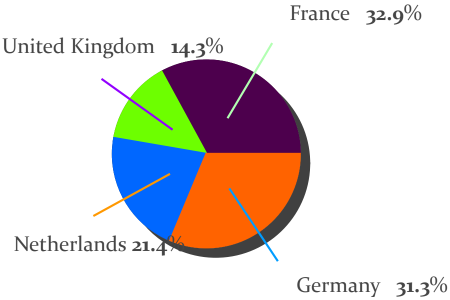
Some JJS Statistics 1/2



$\Longrightarrow$ On average I used more than 2000 CPUs 24/7

Some JJS Statistics 2/2

Number of running jobs: 2 119



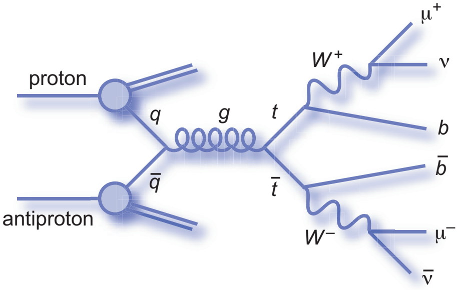
Number of successful jobs for the last campain of job submission:

More than **99%** of successful jobs after my modifications ($< 80\%$ for WMS).

This is the consequence of the JJS job manager policy which resubmit jobs on other remote computing elements if their transfer failed.

Most of these problems come from maximum CPU time exceeded (the correct amount of needed CPU time is quite hard to determine.)

The Measurement of the Top Quark Mass with the Matrix Element Method in the Dilepton Channel



Description of the Matrix Element Method

The aim: Create a likelihood function depending on the top quark mass which have a maximum for its most probable value

The method: We calculate the probability that an event is either signal or background as a function of the top quark mass:

$$P_{\text{evt}}(\mathbf{x}, m_t, f_s) = f_s P_s(\mathbf{x}, m_t) + (1 - f_s) P_{\text{bkg}}(\mathbf{x})$$

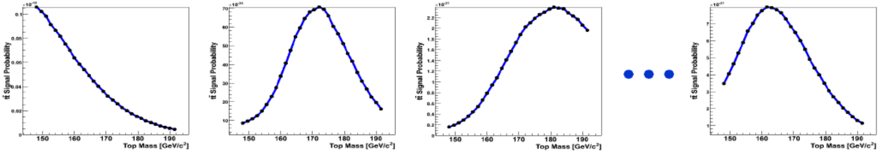
$$P_{\text{s,bkg}}(\mathbf{x}, m_t) = \underbrace{\frac{1}{\sigma_{\text{acc}}}}_{\text{Normalization}} \int \underbrace{\mathrm{d}^6\sigma_{\text{s,bkg}}(\mathbf{y}, m_t)}_{\substack{\text{differential cross-section:}\\ \text{matrix element LO}\\ \text{6-body phase space}}} \times \underbrace{W(\mathbf{x},\mathbf{y})\mathrm{d}y}_{\substack{\text{transfer function:}\\ \text{probability for}\\ \text{parton parameter } \mathbf{y}\\ \text{to be measured as } \mathbf{x}}} \times \underbrace{f(q_1)f(q_2)}_{\substack{\text{parton}\\ \text{distribution}\\ \text{function}}} \mathrm{d}q_1 \mathrm{d}q_2$$

$P_{\text{s,bkg}}$	signal or background probability per event
f_s	signal fraction
$\mathbf{x}$	measured parameters (angles, energy)
m_t	tested top quark mass
q_1, q_2	fraction of momentum of initial parton

$\Longrightarrow$ The integration is done on the remaining degrees of freedom (under-constrained kinematics) in n-dimensional space by Monte Carlo method.

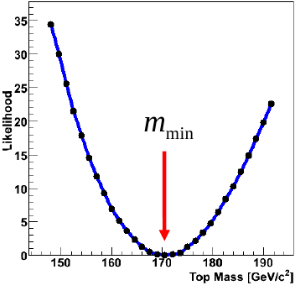
The Matrix Element Method in Practice

We calculate the probabilities $P_{\text{evt}(\text{x},m_t,f_s)}$ for each individual event:



Then we compute the likelihood L and get the $-\ln$:

$$\mathcal{L}(\mathbf{x_1},\mathbf{x_2}\ldots\mathbf{x_n},m_t,f_s) = -\ln L = -\ln\prod_{i=1}^{n}P_{\text{evt}}(\mathbf{x_i},m_t,f_s)$$



The best estimate of the top mass is then determined by minimizing the $-\ln L$.

The statistical error can be estimated from:

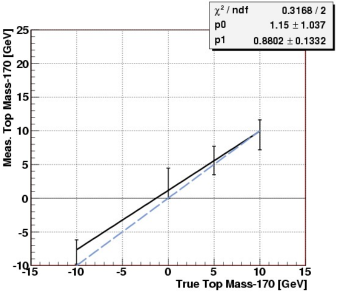
$$\ln L(m_{\text{min}}) - \ln L(m_{\text{min}} \pm \sigma) = \frac{1}{2}$$

The Calibration of the Method

The optimum of the likelihood is not exactly the mass used in the generator.
$\Longrightarrow$ Calibration of the method

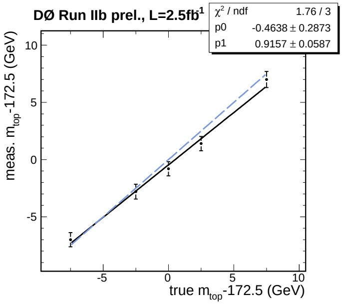
Generation of the calibration:

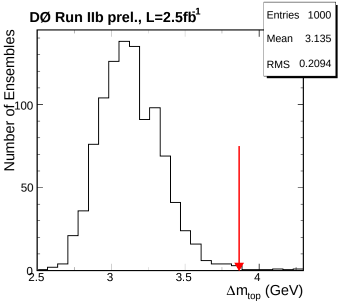
- Monte-Carlo events are generated with different top quark mass
- Ensembles are created with randomly chosen events following the number of expected signal and background events in the data sample
- In each ensembles the optimal value of the likelihood (m_{min}) is calculated
- The average of the m_{min} found in each of the ensembles is computed and compared to the input mass

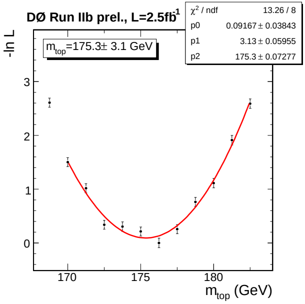


Sample of calibration curve for p17 ($\mu\mu$ channel)

Application of the Method for the $e\mu$ Channel (Moriond 2009)



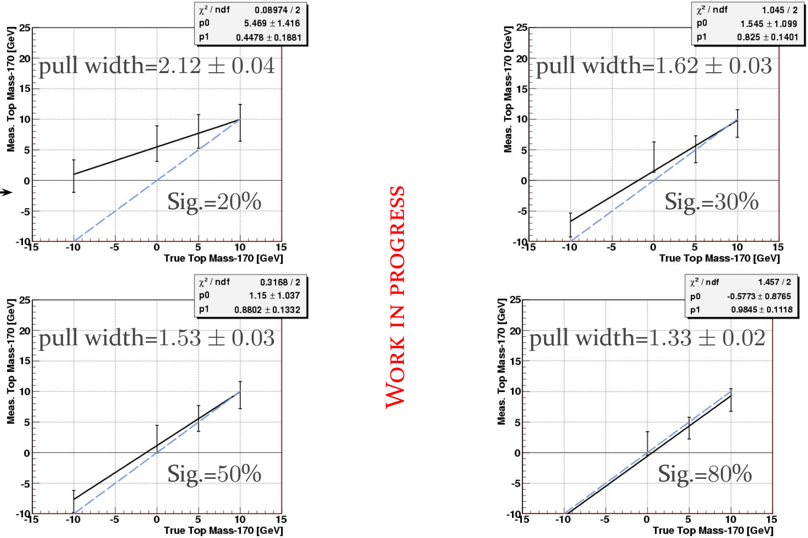




Work in Progress for the μμ Channel: Calibration for Different Ratio of Signal and Background

Current basic selection

	Z, MC	Diboson MC	Number of fake muon events	$t\bar{t} \to e\mu jj$ MC (cross-sec 7.45 pb, m_t = 172 GeV)	Expected N of events	N of events
MET $\geq$ 40 GeV	$80.0^{+1.0}_{-1.0}$	$2.7^{+0.1}_{-0.1}$	$36.6^{+6.6}_{-6.6}$	$20.4^{+0.1}_{-0.1}$	$139.7^{+6.7}_{-6.7}$	128



Conclusion

- We now have a solution to submit jobs on the LHC Computing Grid
- This solution has already been used to integrate events for the Moriond conference
- It is currently used to measure the top quark mass in the $\mu\mu$ channel
- Some improvement need to be done in the JJS code yet
- It will be available to the DØ collaboration very soon
- We are working on the $\mu\mu$ selection