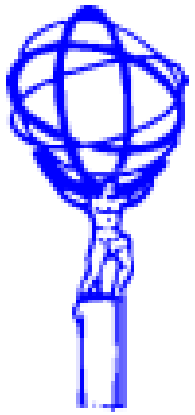
Optimization of Matrix Element Method using GPUs



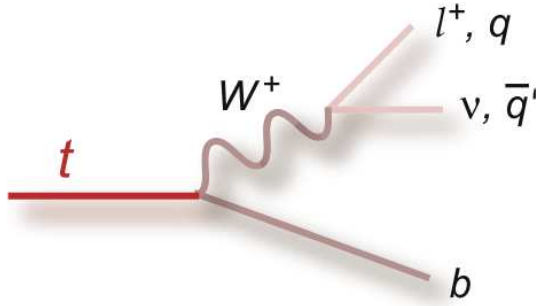
Robert Harrington

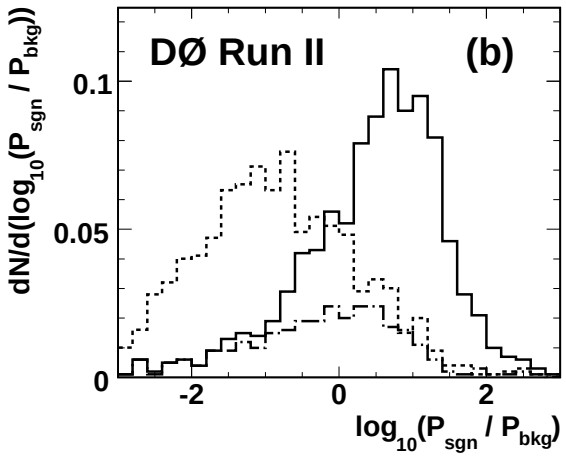
Overview

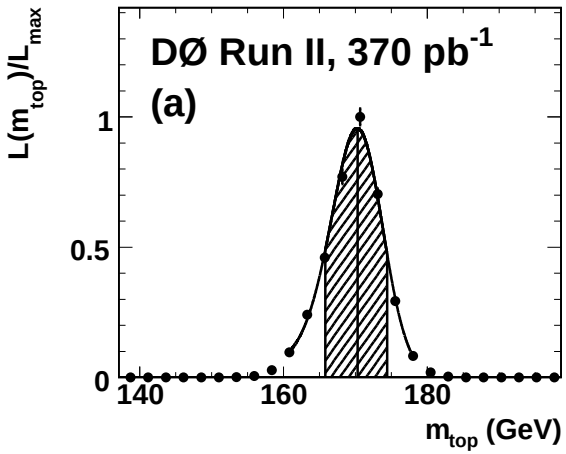
- Description of matrix element method
- DØ vs. LHC
- MadGraph studies using GPUs
- Prospects and conclusions

Top Mass Measurement using ME Method

- For every event, calculate probability to be top decay, $P_{sgn}(m_t, JES)$, and probability to be background, P_{bkg}, using differential cross-sections.
- Jet-parton combinations: 4!=24 ways to match 4 jets to 4 quarks in final state.
- For P_{sgn}, reduced to 12 by using an average of P_{sgn} with 2 quarks from hadronic W decay interchanged.
- P_{sgn} calculated for each of a number of m_t hypothesis.
- P_{sgn} and P_{bkg} also calculated for a number of jet energy scales, thus allowing a smaller JES systematic.

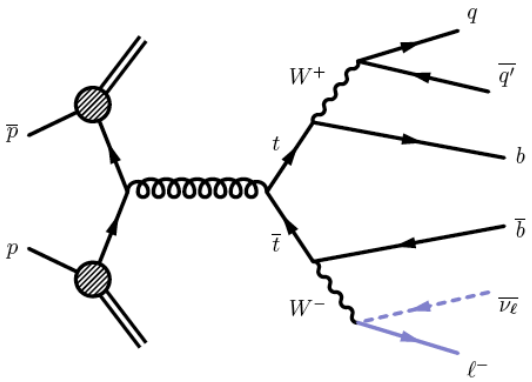






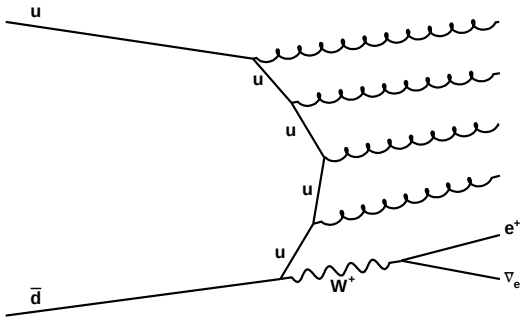
CPU consumption at DØ

P_{sgn}



- 12 calculations done for each m_t and JES hypothesis.
- At DØ 30 m_t and 21 JES values were used $\rightarrow 12 \times 30 \times 21 = 7560$ P_{sgn} per event,
- 1000 events took 5-15 hours for each m_t/JES pair on a Pentium IV 2.5 GHz CPU.
- 40-110 CPU-hours/event

P_{bkg}



- Calculation done using many matrix elements calling VECBOS with a simplified integration over jet energies.
- Calculation not done for different JES values due to insufficient sensitivity to JES.
- 40 events took 15-30 hours on Pentium IV 2.5 GHz CPU.
- 22.5-45 CPU-minutes/event

Comparison of DØ and LHC

	DØ	LHC
theoretical $\sigma_{t\bar{t}}$	$7.27^{+0.76}_{-0.85}$ pb (m_t=172.5 GeV)	165^{+11}_{-16} pb (m_t=172.5 GeV)
$\mathcal{L}$	$10^{32} cm^{-2} s^{-1}$	$10^{33} cm^{-2} s^{-1}$
$\int \mathcal{L}$ dt	400 pb^{-1} (2006)	160 pb^{-1} (EPS analysis)
number of events	175 l+jets	2000 l+jets
Matrix element $\mathcal{M}$	85% $q\bar{q}$ 15% gluon fusion	15% $q\bar{q}$ 85% gluon fusion
$\mathcal{M}$ calculation	1 LO diagram analytical calculation (incl. spin correlations)	3 LO diagrams no analytical calculation ISR/FSR more important
CPU time	7100-14100 CPU-hours	4-8 $\times 10^5$ CPU-hours

➲ The matrix element method will be much more challenging at the LHC!

(These times are probably a little better now due to improvements in CPU.)

Matrix Element Method: A Closer Look

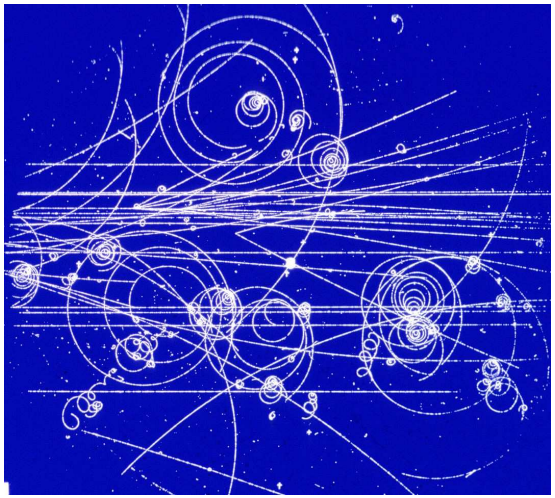
$$ln\ L = \sum_{i=1}^{N} ln\ \left(f_{sgn}\ P_{sgn}(m_t, JES) + (1 - f_{sgn})P_{bkg}\right)$$

P_{sgn} and P_{bkg} are proportional to differential cross-sections calculated using Fermi's Golden Rule -

$$P = \frac{1}{\sigma_{tot}} \sum_{flavors} \int_{x_1} \int_{x_2} dx_1 dx_2\ f(x_1) f(x_2)\ \mathcal{F}\ |\mathcal{M}|^2\ d\Phi_n\ W(x, y)$$

- dx_1, dx_2: incoming parton energy fractions
- $f(x_1), f(x_2)$: parton distribution functions
- $\mathcal{F}$: flux factor
- $|\mathcal{M}|^2$: scattering amplitude
- $d\Phi_n$: phase space differential
- $W(x, y)$: transfer function (parton $\leftrightarrow$ jet energy)

Speeding up calculation



- Many types of physics-motivated approximations possible to speed up integration (at the expense of accuracy):
  - Simplify/skip integration over jet/lepton energies.
  - Reduce number of Feynman diagrams.
  - Tighter event selection (incl. b-tagging) to reduce number of events.
  - Ignore some jet-parton combinations.
- Choice of integration variables has huge impact on integration time using importance sampling.
- DØ has already shown an improvement since the 2006 measurement. At LHC, we can afford to have a really tight event selection.
- Parton distribution functions use PDFLIB Fortran subroutines, not much improvement possible.
- Transfer functions: likely some room for improvement through parametrization.
- Parallelization of integration over phase space.

Importance sampling with GPUs

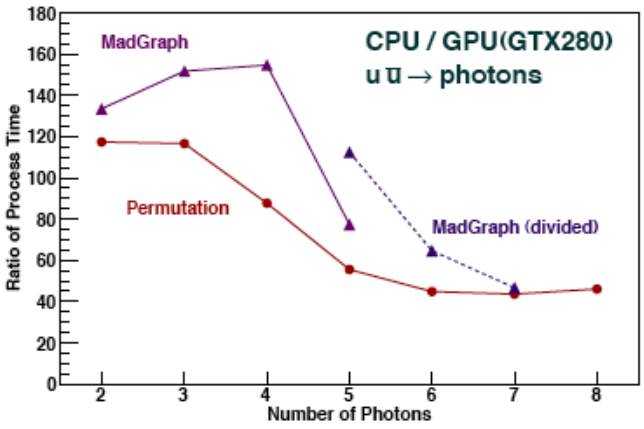
- In importance sampling, points in phase space chosen at random, and the integrand is calculated at that point in phase space.
- This is done for a large number of points in phase space for the 1st iteration.
- After the first iteration, regions of phase space given weights based on contribution of each region to total integral. Higher contributions $\leftrightarrow$ higher weight.
- For subsequent iterations, weights used for selection of phase space points.
- The integrand calculations are usually done sequentially on CPUs, but there's no reason why it has to be done this way!
- Following approach used with GPUs:
  1. Initialize program.
  2. Generate random numbers on CPU, transfer to GPU.
  3. Using random numbers, determine helicities and momenta of incoming and outgoing partons, compute squared amplitudes on GPU.
  4. Transfer momenta, helicities, and squared amplitudes back to CPU.
  5. Sum all values to obtain the total cross-section.

Using GPUs

Useful to look at cross-section calculations done using Monte Carlo techniques:

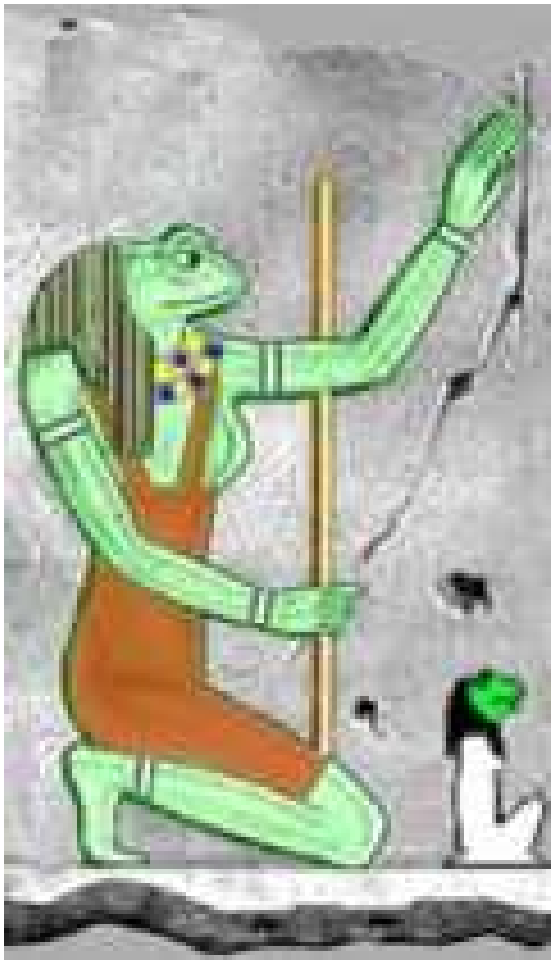
$$\mathrm{d}\sigma^{n\gamma} = \sum_{flavors} \int_{x_1} \int_{x_2} dx_1 dx_2 \; f(x_1) f(x_2) \; \mathcal{F} \; \frac{1}{2\hat{s}} \frac{1}{2^2} \frac{1}{3^2} 3 \sum_{\substack{\lambda_1,\cdots,\lambda_n \\ \sigma_1,\sigma_2}} \left| \mathcal{M}^{\lambda_1,\cdots,\lambda_n}_{\sigma_1,\sigma_2} \right|^2 \frac{1}{n!} \mathrm{d}\Phi_n$$

- K.Hagiwara *et al.*[1] used GPUs to speed up determination of cross-section for $pp \longrightarrow n\gamma$ using MadGraph for 2 to 8 photons.
- Squared matrix element amplitude: sum of squares of $n!$ matrix elements.
- Replaced Fortran-based HELAS subroutine with CUDA-based HEGET subroutines.
- Able to achieve 45-150 times improvement in computing by using GPUs.
- GPU: GeForce GTX280 with 30 multiprocessors, 240 cores, 1.3 GHz
- CPU: Core2Duo 3 GHz with 64-bit Fedora 8



[1] Hagiwara K., Kanzaki J., Okamura N., Rainwater D. & Stelzer T., European Physical Journal C 66, 477-492 (2010).

MadGraph with HEGET



- Two versions of code written in HEGET (**H**ELAS **E**valuation with **G**PU **E**nhanced **T**echnology) to compute amplitudes:
  - FORTRAN code from MadGraph converted to HEGET.
  - Handwritten program using permutations of final state photons.
- Double vs. single precision: double precision version of HEGET runs 3-4 times slower, but results agree with single precision version so perhaps single precision is ok.
- Unrolling loops
  - Scattering amplitude calculated for each permutation in a while loop.
  - Unfolding gives a factor of 3 improvement.

Expected improvement for matrix element method

- P_{sgn} and P_{bkg} integrations both have 6 dimensions for final state partons + 2 for incoming partons.
- The $pp \rightarrow 6\gamma$ cross-section integral has the same dimensionality (6+2), so the performance factor should be similar.
- The improvement using GPUs for $n = 6$ was about a factor of 60, so we expect a similar improvement for the P_{sgn} and P_{bkg} calculations.
- Using CPUs, expect 200-400 CPU-hours per event. With 60$\times$ improvement, we can achieve 3-7 hours per event.
- Total computing time for 160 pb^{-1}: 4-8 $\times 10^5 \rightarrow$ 7-14 $\times 10^3$ CPU-hours.
- This is close to CPU consumption for 2006 DØ top mass measurement with much higher statistics!

Conclusions

- Studies already done into speeding up MadGraph with GPUs lay the foundation for improving a matrix element method.
- For simple $pp \longrightarrow n\gamma$, factor of at least 60 improvement.
- Expect matrix element method for top mass measurement LHC to have a similar improvement.
- 7-14 $\times 10^3$ CPU-hours to analyze 160 pb^{-1} is a lot, but certainly possible.
- Might be harder to sell a top mass measurement today than it was in 2006, so we'll probably have to do better than this.
- Some specific issues to worry about:
  - Complicated amplitude calculation due to predominance of gluon fusion diagrams.
  - LO vs. NLO, need to include ISR/FSR.
  - Transfer functions.

Conclusions (cont'd)

- Implementations already exist and/or are being developed using CPUs, so some thought can go into implementation design to eventually accomodate GPU-assisted versions.
- Of course, the method can be applied to other interesting and exciting measurements!


That's all Folks!