Computing Frontier: Perturbative QCD
(CpF T4)

Stefan Höche and Laura Reina (Conveners)
Markus Wobisch (Observer)

Snowmass Community Meeting

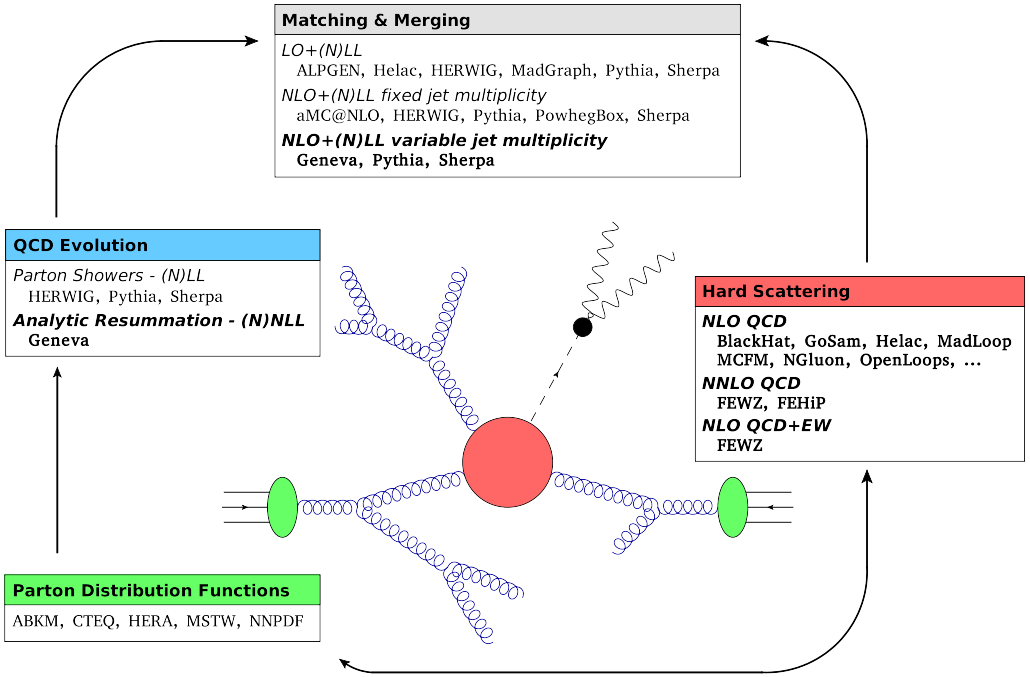
Minneapolis, July 30, 2013

Thanks to: C. Bauer, Z. Bern, R. Boughezal, J. Campbell, L. Dixon, T. Gehrmann, J. Kanzaki, A. Mitov, P. Nadolsky, F. Olness, M. Peskin, F. Petriello, S. Pozzorini, F. Siegert, D. Wackeroth, J. Walsh, C. Williams

and in particular

Lali Chatterjee and Larry Price (DOE), Richard Gerber (NERSC), Tom LeCompte (ANL), Salman Habib (ANL), Richard Mount (SLAC)

Broad impact of Perturbative QCD on collider physics



- interpreting LHC data requires accurate theoretical predictions
- complex SM backgrounds call for sophisticated calculational tools
- higher order QCD(+EW) corrections mandatory

This effort could greatly benefit from:

- unified environment for calculations/data exchange
- adequate computational means to provide accurate theoretical predictions at a pace and in a format useful to experimental analyses
- extensive computational resources to explore new techniques

As pQCD component of the Computing Frontier we have set:

- **Short term goals**
  - ▷ provide collider experiments with state-of-the-art theoretical predictions;
  - ▷ make this process automated/fast/efficient;
  - ▷ facilitate progress of new ideas and techniques for cutting-edge calculations (NLO with high multiplicity; NNLO).
- **Long term goals**
  - ▷ take advantage of new large-scale computing facilities and existing computer-science knowledge;
  - ▷ work in closer contact with computing community to benefit from pioneering new ideas (GPU, Intel Phi, programmable networks, ...).

We have explored available options and provided some proofs of concept

More specific charges:

- **Provide summary of current computing needs**
  - ▷ Available tools and their CPU & storage requirements
  - ▷ Prospects for exploiting these tools beyond their original scope
  - ▷ Increased computing and storage capacity $\rightarrow$ increased potential?
  - ▷ Can we facilitate (semi-)automatic production of results?
- **Assess best infrastructures needed in the future**
  - ▷ What is the role of parallel computing?
  - ▷ What can be gained from consolidating resources?
  - ▷ Are there limitations in the software environment?
  - ▷ ...

NLO

Status

- Conceptual/technical challenges largely met
- New & old techniques for one-loop QCD implemented in several (public) codes, matching
  → One-Loop Providers and
  → Monte-Carlo event generators
- Interface with Parton Shower Monte Carlo at NLO available

Issues to consider

- Availability of codes, grade of automation, expandability, versatility (e.g. implementation of cuts, jet vetos), user friendliness
- Can improved computing help to better exploit existing tools?
  (e.g. provide power to run w/ different parameters or cuts
  provide storage needed for large event files / ntuples)
- How do automated codes perform with increasing number of particles?

Resource requirements

Prototype cutting-edge NLO parton-level results for LHC physics:
Blackhat+Sherpa

Process	Requirements	
	CPU [core h]	Storage [GB]
$pp \to W^{\pm} + 5jets$	600,000	1,500
$pp \to W^{\pm} + 4jets$	100,000	200
$pp \to Z + 4jets$	200,000	200
$pp \to Z + 3jets$	50,000	100
$pp \to 4jets$	200,000	150

Required Monte-Carlo accuracy $\to$ meaningful comparison with data

Combining One-Loop-Providers+Monte-Carlo event generators all $2 \to 2, 3, 4$ processes relevant for LHC physics can be made available in a common framework:

$\hookrightarrow$ NLO repository available for multiple runs

Beyond parton-level: NLO+fully exclusive event generators (using Sherpa)

Process	N_{jet}		CPU [core h]
	NLO	LO	
$pp \to W^{\pm} + jets$	≤ 2	≤ 4	100,000
$pp \to h + jets$	≤ 2	≤ 3	150,000
$pp \to t\bar{t} + jets$	≤ 1	≤ 2	250,000
$pp \to l\bar{\nu}\bar{l}'\nu'$	≤ 1	≤ 2	50,000

- combine multiple NLO-matched calculations for varying jet multiplicity
- produce inclusive event samples which can be reduced to NLO-accurate predictions at arbitrary jet multiplicity
- very demanding since rely on high-multiplicity NLO calculations

$\hookrightarrow$ More examples in CpF T4 Report

NNLO

Status

- State of the art is $2 \to 2$ processes with massive particles (e.g. $t\bar{t}$ hadroproduction) or $2 \to 1$ processes fully differentially.
- Still big challenges to be met in computing both two-loop corrections and double-parton emission (still building tools).

Questions to be addressed

- How are NNLO calculations evolving: Are they going to be mainly analytical or mainly numerical in nature?
- Can computational issues and bottlenecks be identified already? How do we expect the need of computational power to scale?
- Are there intrinsically different computational issues at NNLO compared to NLO?

Resource requirements

Process	Requirements CPU [core h]	CPU clock [GHz]
$pp \to W/Z$	50,000	2.67
$pp \to H$	50,000	2.67
$pp \to t\bar{t}$	1,000,000	2.27
$pp \to$ jets (g only)	85,000	2.20
$pp \to H$+jet (g only)	500,000	2.67

Required Monte-Carlo accuracy $\to$ meaningful comparison with data

- methods and techniques still being developed $\to$ more resources could boost this phase
- needed resources are very process/method dependent
- at NNLO resources for PDF development not marginal $\to$ see Report.

Computational Tools

- **Parallelization**
  - ▷ Multi-Threading
    - → Communication across processor cores (CPU/GPU)
    - → Shared memory between all threads - implicit communication
    - → Not scalable → reduction of processing time by at most # of cores
  - ▷ Message Passing Interface (MPI)
    - → Communication across processor cores or computing nodes
    - → No shared memory between threads - communicate explicitly
    - → Scalable → "arbitrary" reduction of processing time
- **Distributed Computing**
  - ▷ Local Computing Clusters
    - → Small-scale local batch processing
    - → Large-scale parallel computing (SC centers, e.g. NERSC)
  - ▷ Open Science Grid
    - → Capable of absorbing peak loads, impossible at single sites
    - → Details of resource allocation hidden from user
    - → MPI capable, but no inter-node communication yet

Parallel vs serial computing, an example

MC simulations/ NLO pQCD calculations can be split into

- Integration steps
  - ▷ Determine total cross section and maximum for MC simulation
  - ▷ Use adaptive MC integrators to reduce variance
  - ▷ store results in form of weight factors/grids
- Event generation step
  - ▷ Use weight factors/grids to increase efficiency
  - ▷ Produce full events instead of cross sections only (parton showers, hadronization, ...)

Integration step is domain of High Performance Computing (HPC)

Optimizing resource usage could mean

↪ Integration performed in parallel (HPC center?)

↪ Event generation distributed (Open Science Grid?)

Study on High Performance Computing for HEP Theory

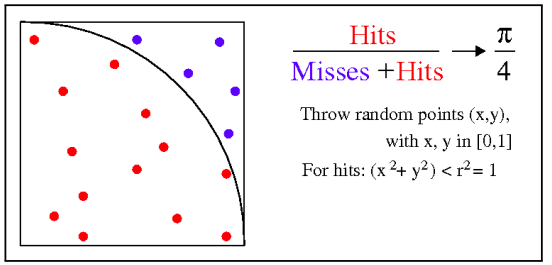
- Strongly supported by DOE Office of Science
  Dedicated allocation (10^6 CPU hrs) at NERSC (National Energy Research Scientific Computing Center) for case studies.
- Drafted white paper "The computing needs of theoretical high energy physics at the Energy Frontier" with focus on
  - ▷ pQCD at NLO and NNLO
  - ▷ New physics searches

  [Bern, Boughezal, Campbell,Christensen, Dixon, Han, Hewett, Höche, Petriello, LR, . . .]
- Tutorial on line on CpF pQCD home page
- Talk+Tutorial presented at the East Coast EF meting (BNL, April 2013)
- Follow up at Loopfest XII (Tallahassee, May 2013)
- Talk+Discussion at Les Houches Workshop (June 2013)

HPC Tutorial at EF East Coast meeting

http://snowmass2013.org/tiki-index.php?page=HPC+Tutorial+at+BNL

Snowmass on the Mississippi a.k.a CSS 2013



Log in

Quick Links

- **TWiki registration**
- **Pre-meetings**
  - Community Planning Meeting
  - All pre-Snowmass Meetings

Groups

- Energy Frontier
- Intensity Frontier
- Cosmic Frontier
- Frontier Capabilities
- Instrumentation Frontier
- Computing Frontier
- Education and Outreach
- Theory Panel

Google Search

snowmass2013.org
WWW

HPC Tutorial at BNL

A tutorial on High Performance Computing for HEP Theory will take place at the Energy Frontier East Coast Meeting on Thursday, 04/04, 12:30-2:00pm in Room C. It is part of the discussion on how to best use HPC facilities for HEP theory.

If you plan to attend, please sign up for a NERSC computing account beforehand by sending an email to shoeche@slac.stanford.edu. You can use this account after the tutorial for HPC studies.

The tutorial will cover simple MPI / OpenMP programming and an example for using MPI to compute NLO cross sections efficiently. Please also plan to attend the general session on HPC for THEP on Wednesday, 04/03, 2:00-3:00pm in Room B.

USEFUL LINKS

Getting started at NERSC
NERSC Tutorials

INTRODUCTION

For details on the HPC for THEP project please refer to its home page. Some remarks on t
in HEP are also given in this talk.

The tutorial will exemplify the relatively simple structure of parallel programs compared t
code. Participants should indicate their availability for exploratory studies in the course of
chart a roadmap for the use High Performance Computing in HEP theory with the help of

INSTRUCTIONS

Log on to NERSC and switch the compiler suite to gcc

```
ssh <username>@hopper.nersc.gov
module unload PrgEnv-pgi
module load PrgEnv-gnu
module load training
```

Copy the examples using

```
cp -r $EXAMPLES .
```

Log on to a mom node (a service node that is used to parse batch jobs and launch paralle
24 cores available for your tests. Sessions time out after 30 minutes.

```
qsub -IV -q interactive -l mppwidth=24
```

Simple C++ MPI program

```
// attach to MPI
MPI::Init(argc,argv);
// get size of and rank in MPI environment
int size = MPI::COMM_WORLD.Get_size();
int rank = MPI::COMM_WORLD.Get_rank();
// initialize random number generator
srand(rank);
// hit it!
double n = 1.0e6, mysum = 0.0;
for (double i = 0.0; i<n; ++i) {
  double x = rand()/(double)RAND_MAX;
  double y = rand()/(double)RAND_MAX;
  if (x*x+y*y<1.0) ++mysum;
}
// collect results
n*=size;
double sum;
MPI::COMM_WORLD.Reduce(&mysum,&sum,1,MPI::DOUBLE,MPI::SUM,0);
if (rank == 0) {
  // compute final answer
  double pi = 4.0*sum/n;
  double sig = 4.0*sqrt((sum/n-sum*sum/n/n)/(n-1.0));
  std::cout<<"\\pi = "<<pi<<" +- "<<sig<<std::endl;
}
// detach from MPI
MPI::Finalize();
```

Facilities available

- Cray XE6™ "Hopper" at NERSC
  24 AMD Opteron™ 2.1 GHz cores per node (153,216 total cores)
  32/64 GB RAM per node (6,000/384 nodes)
  Cray Gemini 3D Torus Network
- Cray XK7™ "Titan" at OLCF
  16 AMD Opteron™ 2.2 GHz cores per node (299,008 total cores)
  32 GB RAM per node (all nodes)
  NVidia® K20 GPU accelerators (18,688 total GPUs)
  Cray Gemini 3D Torus Network
- IBM® BlueGene®/Q test system "Vesta" at ALCF
  16 1.6 GHz PowerPC® A2 cores per node (32,768 total cores)
  16 GB RAM per node (all nodes)
  IBM 5D Torus Network
- Open Science Grid (OSG)

Experience gained so far

- porting on the two Cray systems more convenient (standard Linux environments), but standard software available on all three systems
- MPI communication implemented into a representative Monte-Carlo event generator framework (Blackhat+Sherpa, "One-loopers"+Sherpa)
- used in cutting-edge calculations, e.g. $pp \to W + 5$ jets
  - ▷ observed weak scaling up to 8,192 cores and strong scaling up to 1,024 cores on "Hopper" (NERSC) and "Titan" (OLCF)
  - ▷ "Vesta" (ALCF) has lower clock frequency, need to increase number of cores by a factor 2.2: weal scaling tested up to 16,000 cores.
- MPI more efficient than multi-threading for current Monte-Carlo applications. Possible future optimizations.
- Explored MPI on OSG running small-scale HPC jobs. Excellent usability. Only limit: number of cores accessible still limited by the number of cores per node (between 4 and 64).

- Preliminary: parallel computing using accelerators (GPUs) tested on benchmark process $u\bar{d} \to W^+ n$ gluons (J. Kanzaki)
  (uses BASES/SPRING package, by S. Kawabata)
  Ratios of CPU vs GPU execution times (gain) very promising:

n-gluons	integration (BASES)	generation (SPRING)
0	95	24
1	84	44
2	67	70
3	39	>1000
4	18	n.a.

GPU: NVidia Tesla C2075 GPU with CUDA™ 4.2
CPU: was an Intel Core i7 2.67 GHz

  - ▷ caveat: only one CPU core
  - ▷ increasing gain with n due to parallel nature of underlying algorithm

Main results and recommendations

- Resource requirements were determined for the calculations of prototype NLO and NNLO calculations.
- Different HPC environment were tested and their suitability for pQCD calculations assessed.
- Repository of codes for LHC physics started at NERSC.
- Access to HPC resources will be very beneficial for
  - ▷ making existing calculational tools available to extensive experimental studies in a coherent well-tested framework, without depending on local computer and man power
  - ▷ providing enough resources for new cutting-edge calculations, both for running and development
- Local resources will still be vital for prototyping and development and could be effectively integrated in distributed systems (e.g. OSG).