Enabling the ATLAS Experiment at the LHC for High Performance Computing

Masterarbeit an der philosophisch-naturwissenschaftlichen Fakultät der

Universität Bern

vorgelegt von

Michael Hostettler

2015

Leiter der Arbeit

Prof. Dr. A. Ereditato

PD Dr. S. Haug

Albert Einstein Center for Fundamental Physics

Laboratorium für Hochenergiephysik
Physikalisches Institut

CERN-THESIS-2015-391
//2015

Abstract

In this thesis, I studied the feasibility of running computer data analysis programs from the Worldwide LHC Computing Grid, in particular large-scale simulations of the ATLAS experiment at the CERN LHC, on current general purpose High Performance Computing (HPC) systems. An approach for integrating HPC systems into the Grid is proposed, which has been implemented and tested on the „Todi" HPC machine at the Swiss National Supercomputing Centre (CSCS). Over the course of the test, more than 500000 CPU-hours of processing time have been provided to ATLAS, which is roughly equivalent to the combined computing power of the two ATLAS clusters at the University of Bern. This showed that current HPC systems can be used to efficiently run large-scale simulations of the ATLAS detector and of the detected physics processes. As a first conclusion of my work, one can argue that, in perspective, running large-scale tasks on a few large machines might be more cost-effective than running on relatively small dedicated computing clusters.

The second part of the thesis work covers a study of the discovery potential for supersymmetry (SUSY) by studying ATLAS events with one lepton, two b-jets and missing transverse momentum in the final state. By using flat-random distributed pMSSM models, I identified some models which could possibly lead to the discovery of SUSY by this specific channel.

Contents

Publications and Presentations

Over the course of this study, I gave and contributed to several talks and posters on the usage of High-Performance Computing for High Energy Physics in general, and for the ATLAS experiment in particular, listed in the following:

- M. Hostettler, S. Haug, P. Fernandez, R. Walker. *Enabling ATLAS for CSCS HPC*, Project Report and CSCS Access Extension Request. March 31, 2014.
- M. Hostettler, S. Haug. *ATLAS on Piz Daint*, Talk in the ATLAS HPC working group. April 22, 2014.
- M. Hostettler, S. Haug. *ARC CE ssh backend*, Talk in the ATLAS HPC working group. May 21, 2014.
- S. Haug, M. Hostettler. *Enabling Large Hadron Collider (LHC) for HPC*, PASC'14 talk. June 2, 2014.
- M. Hostettler, S. Haug. *Using CSCS HPC Resources for LHC ATLAS Data Processing*, PASC'14 poster. June 2, 2014.
- M. Hostettler, S. Haug. *ATLAS on CSCS HPC: Bern efforts and status*, Talk at the CHIPP Computing Board meeting. August 19, 2014.
- M. Hostettler, S. Haug. *ATLAS on CSCS HPC*, Talk at the ERC4HEP meeting on future Grid and HPC integration. August 26, 2014.
- S. Haug, S. Gadomski, M. Hostettler, G. Sciacca. *Enabling LHC searches for the unknown content of the universe*, CSCS CHRONOS Project Proposal for 2015. October 10, 2014.
- M. Hostettler, S. Haug. *Bern/CSCS status and activities*, Talk in the ATLAS HPC working group. December 10, 2014.
- S. Haug, M. Hostettler. *Running on Cray: Status and Thoughts*, Report for the CHIPP Computing Board. January 29, 2015.
- M. Hostettler, A. Filipcic, S. Haug, J. K. Nielsen, S. Raddum, R. Walker. *ARC-CE based HPC integration for ATLAS*. Talk at the ATLAS Software & Computing Week. February 7, 2015.
- M. Hostettler, S. Haug, G. Sciacca. *The ATLAS ARC ssh back-end to HPC*, CHEP'15 poster with proceedings, publication pending. April 2015.
- S. Haug, A. Filipcic, M. Hostettler, R. Walker. *ATLAS computing on the HPC Piz Daint machine*, CHEP'15 poster with proceedings, publication pending. April 2015.

1 Introduction

1.1 Introduction

Processing and analyzing data taken by the detectors at the Large Hadron Collider (LHC) is a major computational challenge, both due to the amount of data to be distributed, and due to the computing time needed to process it. Currently, data processing is based on a hierarchical, worldwide distributed grid computing system, the Worldwide LHC Computing Grid (WLCG). WLCG computing sites are run e.g. by universities or scientific computing centers on computer clusters specifically set up to meet the needs of the LHC experiments. More than one million grid jobs run on the distributed computing sites all over the world on more than 200000 CPU cores.

While the approach of running LHC grid jobs on dedicated, specifically setup computing clusters showed to be reliable during LHC Run 1, it also bars the LHC experiments from using big general purpose supercomputers, also referred to as „High Performance Computing (HPC) systems". Purchasing and operating the dedicated computing clusters requires a major effort from CERN and the experiment collaborations, so getting a share on a few big supercomputers might be a more efficient way for large-scale data processing than running many independent computing clusters. Also, the need for computing resources will increase with increased data yields due to the LHC upgrades already implemented for Run 2 or planned for the future (e.g. the HL-LHC).

In this work, the feasibility of running Monte Carlo simulations for the ATLAS experiment on HPC systems at the Swiss National Supercomputing Centre (CSCS) was studied. In section 2.5, a possible solution for integrating HPC resources in the WLCG is proposed, which was then tested on a small HPC integration system over several months. During these tests, the feasibility was proven, and additional computing power comparable to that of the dedicated computing clusters of the University of Bern was contributed to ATLAS data processing (sections 2.6 and 2.7).

As an additional study, in chapter 3, the sensitivity of a particular decay channel for the discovery of supersymmetric (SUSY) physics beyond the Standard Model was studied. The Standard Model describes our current knowledge on particle physics, including three of the four fundamental forces (strong force, weak force and electromagnetism), all elementary particles currently known, and the Higgs mechanism to allow the particles to gain mass. Although the standard model agrees with most of the current experimental

data to a very high accuracy, it is known that there must be physics beyond, e.g. due to cosmic dark matter (which the Standard Model has no candidate particles for) or due to the fact that Standard Model neutrinos are massless, which is excluded by the observation of neutrino oscillations.

Supersymmetry (SUSY) is a possible property of particles physics models beyond the Standard Model which would solve many of the issues of the Standard Model in a „natural" way. In particular, the „phenomenological Minimal Super Symmetric Model" (pMSSM) was studied, a constraint implementation of a supersymmetric model with certain assumptions to reduce the number of free parameters. The study was focussed on finding the number additional events expected in the ATLAS detector with one lepton and two b-jets ($1l2b$) and missing transverse momentum in the final state if a particular SUSY model was implemented in nature. Based on randomly generated pMSSM models, the fraction of the pMSSM parameter space which could possibly be detected by observing this decay channel was estimated (sections 3.4 and 3.5). The $1l2b$ channel is currently being studied within the ATLAS group at the University of Bern, so my study contributes to a larger topic.

1.2 The Large Hadron Collider

The Large Hadron Collider (LHC) is currently the world's largest particle accelerator located at CERN near Geneva, Switzerland. It accelerates two bunched proton (or lead-ion) beams in their individual beam pipes. Beams cross and are brought into head-on collision in the Interaction Points, where the detectors of the LHC experiments are located. The main experiments are ATLAS [1], CMS [2], ALICE [4] and LHCb [3].

The bunched LHC beams are accelerated through a radio-frequency (RF) system based on superconducting cavities. The top center-of-mass energy reached in 2012 proton-proton collision operation was $\sqrt{s} = 8\,\text{TeV}$. In the arcs of the detector, superconducting dipole magnets generating fields of up to 8 Tesla are used to bend the track of the beam to a closed circle. The LHC is a high-luminosity accelerator: In 2012 proton-proton operation, it provided more than $23\,\text{fb}^{-1}$ of integrated luminosity to the high-luminosity experiments, ATLAS and CMS. This enables good statistics, but also poses major challenges in data recording, management and analysis.

Since the LHC RF system cannot handle arbitrarily low energies, proton bunches must be pre-accelerated and synchronized with the LHC RF system before being injected to the LHC. This is done in multiple steps (Fig. 1.1):

- Proton Source: Protons are generated by ionizing hydrogen gas.
- LINAC 2: In this linear accelerator, the protons are pre-accelerated to 50 MeV.

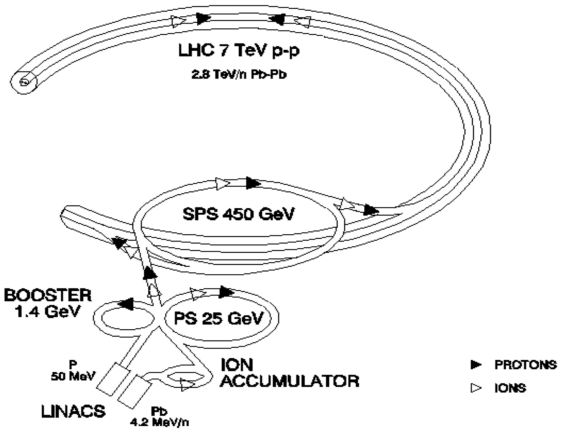


Figure 1.1: The LHC injector complex (taken from [5])

- Proton Synchrotron (PS) and Booster (PSB): The protons are accelerated to 25 GeV, and the beam is bunched with a bunch structure compatible to the LHC RF system. Trains of proton bunches are then injected into the SPS.
- Super Proton Synchrotron (SPS): Accelerates protons to the LHC injection energy of 450 GeV. For the current LHC filling schemes, typically 4 bunch trains from the PS are joined in a SPS batch to be injected to the LHC.
- LHC: Proton bunches from the SPS are accumulated until the final filling scheme for both beams is reached. The beams are then accelerated to reach their final energy (e.g. 4 TeV per beam for a center-of-mass energy of $\sqrt{s} = 8\,\text{TeV}$)

From 2013 to 2015, the LHC is shut down for a major upgrade of both the accelerator itself and its detectors (Long Shutdown 1, LS1). At the time of writing, the CERN accelerator complex is in a re-commissioning phase, with first LHC collisions for physics expected in June 2015. The upgrade will allow the LHC to operate at a center-of-mass energy of up to $\sqrt{s} = 14\,\text{TeV}$[1]. Also, LHC beams will consist of twice as many bunches after the upgrade, increasing the total beam intensity and hence the luminosity. For the first two years of operation, a luminosity of $30\,\text{fb}^{-1}$ is expected.

[1]For 2015, operation at $\sqrt{s} = 13\,\text{TeV}$ is planned

1.3 The ATLAS detector

The ATLAS (A Torodial LHC AparatuS) detector is one of the two high-luminosity general purpose particle detectors at the LHC, the other one being CMS[2]. It is designed to identify the momentum, energy and type of primary and secondary particles created in LHC collisions. In particular, the design was targeted towards detecting the decay channels expected for a Standard Model Higgs boson, which was eventually discovered in 2012 [38].

The detector is of multi-layered cylindrical shape build around the LHC beam pipe (Fig. 1.2). The design is forward-backward symmetric with respect to the beam interaction point. ATLAS consists of three main layers: The innermost layer, the inner detector, are the various tracking systems. The second layer consists of calorimeters to measure the energy of hadrons, electrons and photons. Around the calorimeters, and as end caps at either end of the detector, the muon spectrometer system is installed. The whole detector has a length of ~44 m along the LHC beam pipe, a diameter of ~25 m around the beam pipe, and weights ~7000 t.

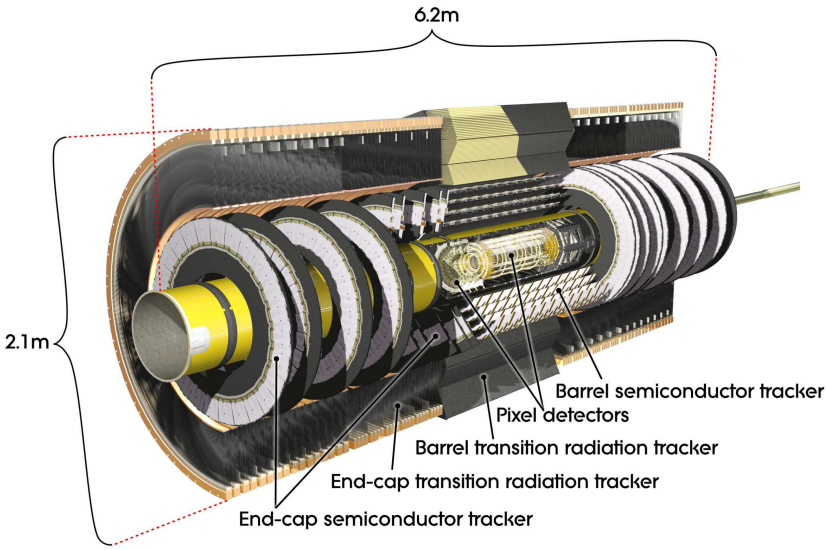


Figure 1.2: Schematic and basic characteristics of the ATLAS inner detector at the LHC (taken from [1]).

The ATLAS inner detector consisted of three tracking detector components, the Pixel Detector, the Semi-Conductor Tracker and the Transition Radiation Tracker. The inner detector is surrounded by a solenoid providing a magnetic field of 2 Tesla to bend the tracks of charged particles, so charge and momentum can be reconstructed.

[2]It is to be noted that ATLAS and CMS are build dissimilarly on purpose, in order to provide results as independent as possible.

During Long Shutdown 1, a fourth tracking system was installed in the inner detector between the beam pipe and the Pixel Detector, the Insertable B-Layer (IBL). It will enhance the detection and tagging of heavy quark jets (b-tagging), the reconstruction of the original vertex (vertexing) due to its increased resolution, and provide additional redundancy for the aging Pixel Detector. The University of Bern was involved in designing, testing and installing IBL and readout components.

Around the inner detector, the electromagnetic calorimeter (ECAL) and the hadronic calorimeter (HCAL) systems are installed. The ECAL absorbs and measures the energy of light particles interacting through electromagnetic interaction, in particular electrons and photons. Based on lead, stainless steel and liquid argon (LAr) cells, it has both a high energy and a high spatial resolution. The HCAL is built around the ECAL and measures the energy of hadrons, which pass through the ECAL. It is based on scintillators.

The outermost system is the muon spectrometer. It consists of large precision tracking chambers in a toroidal magnetic field varying from 3 to 8 Tesla for a very high momentum resolution. A second toroidal magnet system is used for the end cap muon spectrometers.

The measurements taken by all ATLAS systems total to about 1.6 MB of raw data per event after zero suppression [6]. Given the luminosity of the LHC, this would lead to a data stream in the order of petabytes per second. Since it is not possible to process and store this amount of data, ATLAS uses a multi-level trigger system to identify potentially interesting events for permanent storage and offline analysis. The first-level trigger is built directly into the detector electronics and selects about 100000 events per second. The high level triggers run on computing clusters at CERN, and select a few hundred events for permanent storage, leading to a data stream of a few hundred megabytes per second, or petabytes per year [7].

The measured track data for the events selected by the triggers is then distributed in the Worldwide LHC Computing Grid for reconstruction of the physical objects (e.g. Leptons and Jets), which is described in more detail in the next chapter. Along with data produced in Monte Carlo simulations, the reconstructed datasets are then made available for offline analysis by the ATLAS Collaboration.

2 Enabling ATLAS for High Performance Computing

2.1 Motivation

The ATLAS experiment generates petabytes of data per year [7], both data recorded from collisions at the LHC, and simulated data produced for validation and analysis. The production of simulation data consumes about 500 million CPU-hours per year [8], while not being particularly time critical. Currently, these computing resources are provided by computing clusters specifically set up to meet ATLAS' needs.

The goal of this main part of my master thesis was to assess the feasibility of running ATLAS simulations on current High Performance Computing (HPC) systems, also called supercomputers, in a non-intrusive way, i.e. without changing the system configuration or installing local services. Since these systems are usually shared among many users, they cannot be reconfigured to meet particular users needs.

To use HPC systems for ATLAS simulation, a way to submit and manage jobs on an HPC system in an automated way has to be implemented, and access to the ATLAS software used for the simulations has to be provided. Also, the use of accelerators (e.g. Graphics Processing Units) in current HPC should be assessed. In the long term, one could question if HPC systems could even replace dedicated computing clusters for large-scale simulation tasks.

2.2 Cluster Computing

For many tasks in modern scientific computing, a single computer with one or more Central Processing Units (CPUs) is not powerful enough, since the resources (CPU cores, memory, disk space) a machine can use and manage efficiently are limited. Hence, an array of computers, a „computing cluster", can be used for such tasks. Computing clusters typically consist of many individual computers which run the real calculations („worker nodes" or „compute nodes"), one or more management machines, and possibly servers for a network file system shared among all nodes.

The management machines are responsible for queuing and running compute jobs on the compute nodes, distributing the load evenly among all available nodes and possibly enforcing a fair sharing of resources by all users of the cluster. For this purpose, a resource management system (e.g. SLURM [9] or GridEngine [10]) is used. The nodes of a cluster typically run a cluster-targeted distribution of GNU/Linux which allows collective management, e.g. running a maintenance command on all nodes of a cluster without having to log into each node manually.

Compute clusters can be shared among a certain user group or be dedicated to a single purpose. The University of Bern runs a central general-purpose compute cluster (UBELIX) usable by all departments. The Laboratory for High Energy Physics (LHEP) also runs two smaller clusters dedicated to computing for the ATLAS experiment.

2.3 High Performance Computing

High Performance Computing (HPC) systems differ from regular computing clusters in that they are very homogenous, tightly integrated systems with high-bandwidth, low-latency interconnected compute, service and login nodes, providing a Message Passing Infrastructure (MPI) system for inter-node communication. Network communication on the compute nodes is often limited to system-internal communication, internet access is, if allowed at all, very limited. Also, HPC systems are usually shared among hundreds or thousands of users which run self-contained applications without any external dependencies, which are compiled and optimized for the targeted HPC system. Installing software or running persistent services on the systems is usually not allowed.

GPU Computing

A recent development in High Performance Computing is the use of stream processors originally designed as Graphics Processing Units (GPUs) for computing. To accommodate this, NVIDIA, one of the market-leading companies in consumer GPU manufacturing, has developed both a toolchain for compiling code for GPUs (CUDA) and GPUs specifically geared towards a high computing performance.

GPUs follow a massively-parallel architecture, with the latest generation (NVIDIA Tesla K40) incorporating 2880 cores per GPU which share 12 GB of total GPU memory. However, those cores can't be compared to full CPU cores of modern multi-core CPUs, since the capabilities of an individual core are very limited. In particular, the control flow of GPU code is organized in blocks of GPU cores (warps), within which all threads must follow the same control flow. In particular, no conditional branching is possible within a warp, so all threads in a warp which are not executing a particular conditional code are halted until the conditional code finished execution [11]. Also, the throughput when

transferring data from the main (CPU) memory to the GPU memory and vice versa is limited by the PCI-Express bus used to connect the GPUs to the system; hence storing data required for the current operations in the larger main memory is not possible.

This architecture is well suited for applications which process large amounts of independent data in a similar way, e.g. 3D graphics rendering or Grid-based simulations where cells are not coupled during the stepping. For other applications, porting code to efficiently run on GPU is a major challenge. In particular, applications which use more than a few megabytes of memory per thread (with gigabytes shared by several thousand cores), or applications which produce highly-divergent threads e.g. due to step conditions are not well suited to run on GPUs.

The Swiss National Supercomputing Centre

For this work, Cray HPC systems located at the Swiss National Supercomputing Centre (CSCS) were used. CSCS provides a variety of HPC systems to users in all fields of science. A brief overview of the systems considered for this study can be found in Table 2.1. The current flagship system „Piz Daint", a CPU/GPU hybrid Cray XC30 with 5272 compute nodes, is currently Europe's fastest supercomputer.

The system that I primarily used for development and testing of my HPC integration solution was the former integration system, „Tödi"[1]. This is a hybrid CPU/GPU Cray XK7 with 272 compute nodes. The system has ended its regular user operation by September 2014 and is now only available to a few selected users for testing and production. Due to internal CSCS regulations, submitting jobs to the system in an automated manner at a large scale was only allowed thereafter.

[1]For technical reasons, the German umlaut is often neglected, and the system is called „Todi"

System name	*Tödi*	*Piz Daint*	*Piz Dora*	*Monte Rosa*
Model	Cray XK7	Cray XC30	Cray XC40	Cray XE6
Description	Former CPU/GPU development and integration system.	Current flagship hybrid CPU/GPU system.	Flagship CPU-only system.	Former flagship CPU-only system.
Compute node configuration	• 16 core AMD Opteron CPU • 32 GB RAM • NVIDIA Tesla K20X GPU	• 8 core Intel Xeon CPU • 32 GB RAM • NVIDIA Tesla K20X GPU	• 2 x 12 core Intel Xeon CPUs • 64/128 GB RAM	• 2 x 16 core AMD Interlagos CPUs • 32 GB RAM
Number of compute nodes	272	5272	1256	1496
Total number of CPU cores	4352 + 272 GPUs	42176 + 5272 GPUs	30144	47872
Interconnect	Cray Gemini	Cray Aries	Cray Aries	Cray Gemini
Resource Manager / Scheduler	Cray SLURM / ALPS	Cray SLURM / ALPS	Cray SLURM / ALPS	Cray SLURM / ALPS

Table 2.1: Overview of the CSCS HPC systems considered in 2014 for running ATLAS computing.

2.4 The Worldwide LHC Computing Grid

The Worldwide LHC Computing Grid (WLCG) is a distributed infrastructure of computing resources used for production and analysis of data of the LHC experiments. Currently there are more than 170 computing sites (traditionally dedicated or shared computing clusters) organized in the WLCG, processing a total of more than two million jobs every day.

WLCG Tiers

For increased data and job distribution efficiency, the WLCG was designed to be organized hierarchically in four tiers.

- **Tier 0:** There is only one Tier 0 site in the WLCG, the CERN computing center. It is responsible safely storing raw data and performing first pass reconstruction.
- **Tier 1:** These 13 major computing centers around the world are responsible for data handling and distribution to their associated Tier 2 sites. They are connected to CERN via high-bandwidth optical-fiber links, provide tape storage for raw data and a 24/7 support for the Grid.
- **Tier 2:** These sites are typically dedicated clusters run by Universities and scientific institutes. They can store enough data to perform individual analysis tasks, and do their share of simulation data production and data reconstruction, with a certain amount of computing resources pledged to the WLCG. The LHEP at the University of Bern runs its dedicated clusters as a Tier 2.
- **Tier 3:** Tier 3 resources can be individual nodes or shared university clusters without any formal engagement to the WLCG. The UBELIX cluster of the University of Bern is set up as a WLCG Tier 3 resource.

However, the fully-tiered hierarchical structure was loosened due to recent developments beyond the standard Grid cluster computing, including the integration of HPC systems and volunteer hosts through BOINC [12] into the WLCG. Also, with the network bandwidth between the individual Grid sites becoming less limiting than the computing power, the „multicloud" feature was introduced to allow powerful Tier 2 („Tier 2D") sites to get jobs from foreign Tier 1 sites. The HPC test setup I developed run as a Tier 2D site for several months, processing ATLAS production jobs from Tier 1 sites all over the world, even though it didn't have any local storage.

2.5 Grid Middleware

In the different regions of the world participating in the WLCG, different grid middleware stacks are used for Grid Computing and Storage Elements. For Storage Elements, the middleware defines a set of protocols and authentication methods which allow jobs to reference and access the data from the local or shared file system of a storage cluster. On Compute Elements, the middleware provides a common interface for running and managing jobs on different clusters by handling user authentication, accepting jobs and forwarding them to the Local Resource Management System (LRMS) of the cluster.

Most US sites are organized in the Open Science Grid (OSG) collaboration and use the OSG middleware stack. European sites traditionally used the gLite middleware package provided by the Enabling Grids for E-Science in Europe (EGEE) project. Those traditional Grid middlewares are complemented by the ARC middleware developed within the Nordugrid collaboration, which is described more in detail in the next section.

The Advanced Resource Connector (ARC)

For integrating HPC systems into the WLCG, I've implemented an ARC-based front-end which accepts and forwards Grid jobs to run on a remote HPC system. ARC [13] is different from other Grid middleware stacks in that it does not only handle user authentication and job submission to a backing cluster, but also data staging of both input and output files. The ARC approach closely ties a job's requirements to the local cluster capabilities and queue limitations like maximum run time, maximum available memory or software availability, and only submits a job to the Local Resource Management System once all input data is ready. Optionally, ARC can also cache input data files on a local file system of a cluster, so subsequent jobs requesting the same input files can skip the data staging step.

Figure 2.1 shows the typical control and data flow when a job is submitted to an ARC Compute Element (ARC-CE). The clients connects to a HTTPS or GridFTP endpoint at the Compute Element, authenticating himself using a Grid Certificate. Then it sends a job description XRSL (Extended Resource Specification Language [14]), which describes the input files, output destinations, the executable to run, and the requested resources (e.g. number of CPU cores, CPU time, memory, disk space, software and runtime environment). The input files can either be uploaded along with the job description, or be referenced from an external source, e.g. a Grid Storage Element. The ARC Remote Execution Service (A-REX) then decides if it can satisfy this request. A job is only accepted if all requested resources can be provided by the cluster.

When a job is accepted, ARC builds the Session Directory for the job. For this purpose, a directory with an unique name is created and all the files uploaded along with the job are put there. Then, any referenced files from external sources are downloaded into this

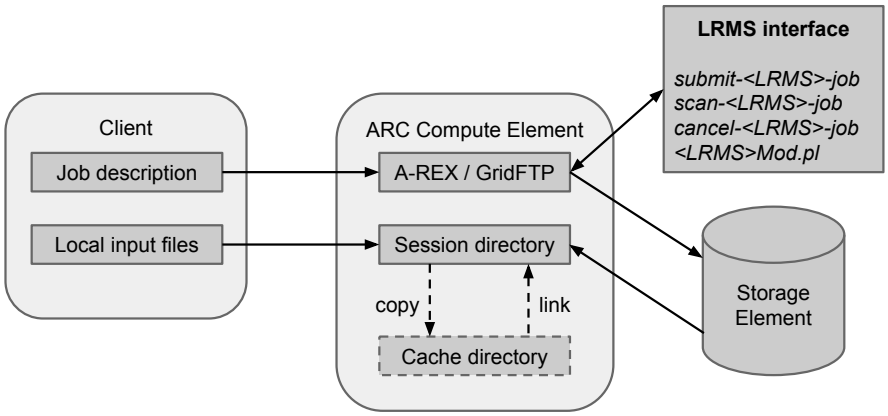


Figure 2.1: Overview of the ARC job information flow. The dashed caching part is an optional feature.

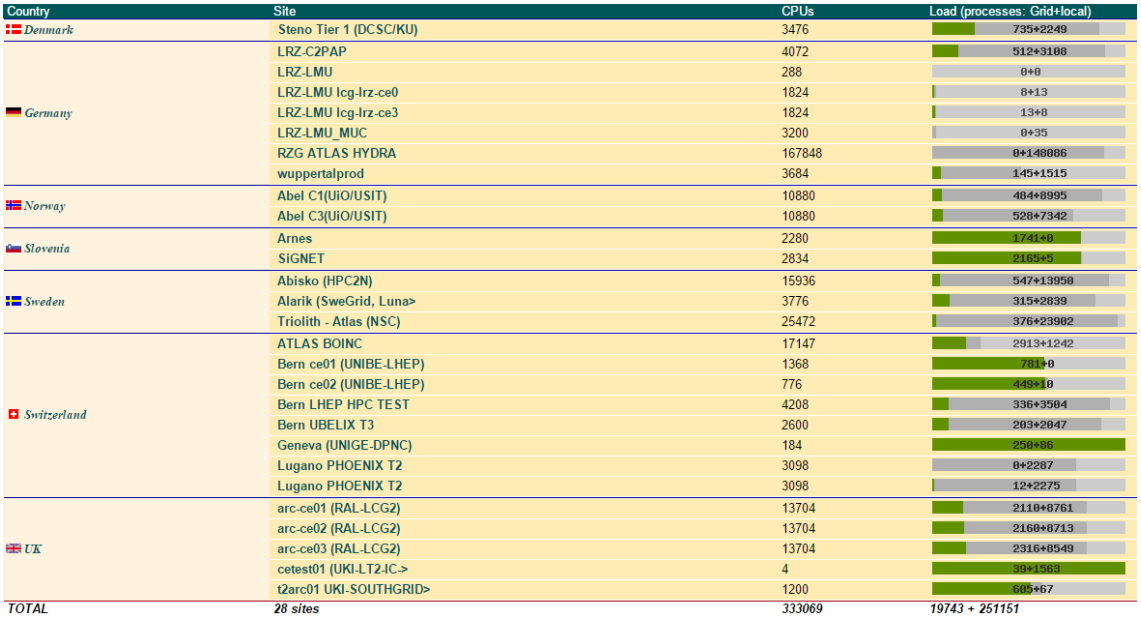
directory (or linked from the local ARC cache in case they are cached from a previous job). Once the session directory is ready, the job is typically sent into the Local Resource Manager System (LRMS) of the computing cluster for execution. When a job from the cluster finishes and an output file destination has been specified in the job description, ARC uploads the requested output files. Job information and the session directory is retained until the client who submitted the job retrieves or deletes it.

An ARC Compute Element also runs services that continuously monitor the configuration and the job state of the local cluster and any running jobs and publishes it to one or more Nordugrid monitoring sites. A screenshot of the main ATLAS Nordugrid monitor [15] including my ARC HPC front-end is shown in Fig. 2.2. Additionally, accounting information for finished jobs is published to the Grid accounting system (EGI-APEL).

2.6 The ATLAS Production and Analysis System

ATLAS computing and data processing is managed by the PanDA (Production and Distributed Analysis) system, which allows users and scientific groups to create tasks or task sets, manages input and output data, and generates jobs for the Worldwide LHC Computing Grid (WLCG). In the following, my work focuses on *production* tasks, i.e. large-scale standardized tasks created to generate events through Monte Carlo simulation, as opposed to custom Analysis tasks defined by individual users.

Monte Carlo production is divided into a chain of individual steps, where the output of one step is used as the input of the next step, as illustrated in Fig. 2.3. All steps are based on transforms defined within the ATLAS Athena software framework:



Country	Site	CPUs	Load (processes: Grid+local)
Denmark	Steno Tier 1 (DCSC/KU)	3476	735+2249
Germany	LRZ-C2PAP	4072	512+3108
	LRZ-LMU	288	0+0
	LRZ-LMU lcg-lrz-ce0	1824	8+13
	LRZ-LMU lcg-lrz-ce3	1824	13+8
	LRZ-LMU_MUC	3200	0+35
	RZG ATLAS HYDRA	167848	0+148886
	wuppertalprod	3684	145+1515
Norway	Abel C1(UiO/USIT)	10880	484+8995
	Abel C3(UiO/USIT)	10880	528+7342
Slovenia	Arnes	2280	1741+0
	SiGNET	2834	2165+5
Sweden	Abisko (HPC2N)	15936	547+13950
	Alarik (SweGrid, Luna>	3776	315+2839
	Triolith - Atlas (NSC)	25472	376+23902
Switzerland	ATLAS BOINC	17147	2913+1242
	Bern ce01 (UNIBE-LHEP)	1368	781+0
	Bern ce02 (UNIBE-LHEP)	776	449+10
	Bern LHEP HPC TEST	4208	336+3504
	Bern UBELIX T3	2600	203+2047
	Geneva (UNIGE-DPNC)	184	258+86
	Lugano PHOENIX T2	3098	0+2287
	Lugano PHOENIX T2	3098	12+2275
UK	arc-ce01 (RAL-LCG2)	13704	2110+8761
	arc-ce02 (RAL-LCG2)	13704	2160+8713
	arc-ce03 (RAL-LCG2)	13704	2316+8549
	cetest01 (UKI-LT2-IC->	4	39+1563
	t2arc01 UKI-SOUTHGRID>	1200	685+67
TOTAL	28 sites	333069	19743 + 251151

Figure 2.2: Screenshot of the ATLAS Nordugrid Monitor [15] on December 19, 2014. My ARC HPC front-end „Bern LHEP HPC TEST" is interfacing the „Todi" HPC system at CSCS and running ATLAS jobs on 336 of its 4208 cores.

1. **Event Generation**: Based on a particle physics model and a set of input parameters (e.g. the collider energy) provided by the creator of the job, a Monte Carlo event generator (e.g. Sherpa, Pythia or Alpgen, depending on the application) is used to generate possible events for the given model and conditions.

2. **Detector Simulation**: The detector response to the events generated in the first step is simulated. The most accurate detector simulator is based on Geant4, a framework to simulate the general case of particles passing through matter. For certain applications, partially parameterized simulations are used, which trade off accuracy for computing efficiency. Full simulations are the most CPU-heavy part of the production chain [16].

3. **File Merging**: Since the detector simulation is more time consuming than any other part of the production chains, event files are split up before being simulated. In this step, the results from the detector simulation are merged back into larger files.

4. **Reconstruction**: Events are reconstructed from the simulated detector output. Through to the simulation and reconstruction steps, it is made sure that the resulting event set takes into account any effects introduced by the detector. The resulting dataset can then be compared to reconstructed data measured by the ATLAS detector.

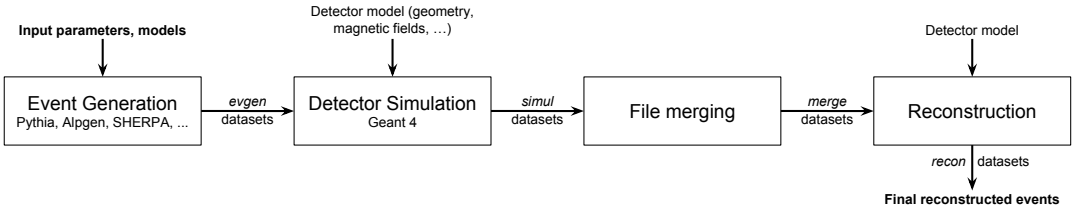


Figure 2.3: ATLAS production workflow, from the generation of events, over the physical simulation of the detector response, to the final event reconstruction.

Integration to the WLCG

For all the tasks set up in PanDA, computing jobs are generated and submitted to the WLCG. PanDA jobs traditionally use a „piloted" approach, where a PanDA Pilot Factory submits a set of dummy jobs (Pilot jobs) to a Grid site. As soon as those jobs are executed on compute nodes, they download the required input datasets from their local Grid Storage Element or a Tier 1, set up the environment and then start the data processing, while continuously sending status updates to PanDA. After finishing the processing, the Pilot registers the output datasets and uploads it to the local Storage Element.

While this approach proofed to be stable with a high throughput, it requires all WLCG sites to provide all the Grid middleware necessary for accessing data on their compute nodes. For computing clusters which are not dedicated to the WLCG, installing Grid middleware on every compute node might be too invasive. Also, the piloted approach renders many of the features of ARC described in section 2.5 useless, since the job requirements are not known at submission time.

In order to combine the advantages of ARC with the flexibility of PanDA, A. Filipcic developed the arcControlTower (aCT) [17]. It plays the role of a PanDA Pilot Factory, but itself fetches the pilot payload from the PanDA server and then submits it to ARC Compute Elements as regular ARC jobs with all the requirements and input files known in advance (Fig. 2.4). Also, all the monitoring and communication with PanDA is handled by the aCT. When run on the destination clusters, the jobs are wrapped by a dummy pilot script which skips all Grid-related operations, so compute nodes don't need to have any Grid middleware installed. Instead, ARC is used for data staging from and to Grid Storage Elements. The aCT has been used for several years on all pure ARC Compute Elements under the Nordic Grid Data Facility Tier 1 (NDGF-T1), including the ATLAS Tier 2 cluster at the University of Bern. It was also used for the HPC integration solution I developed, since installing Grid software on HPC systems is not an option and internet access from HPC compute nodes is usually limited.

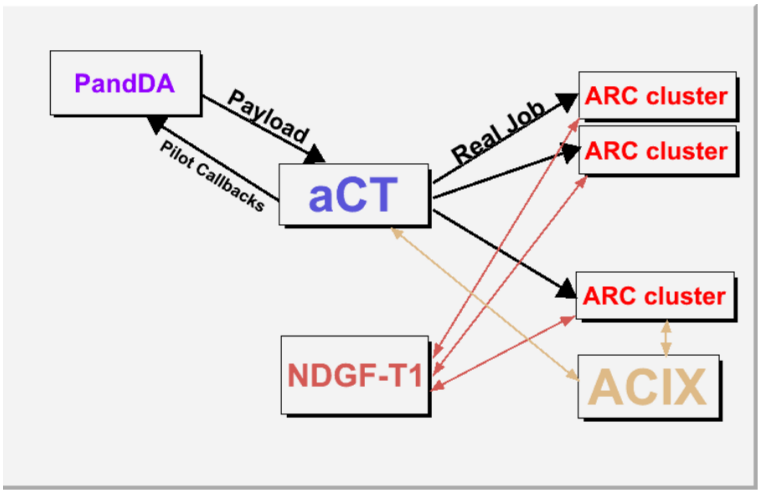


Figure 2.4: The job flow when using ARC and the arcControlTower (taken from [17]).

2.7 HPC and ATLAS

The ATLAS production workflow is quite different from the usual self-contained HPC applications run directly by individual users, and common restrictions of HPC systems (no root access, no possibility to install system-wide software, no internet access from the compute nodes, not being able to run any persistent services) pose different challenges for running ATLAS production jobs. First, ATLAS production jobs depend on the ATLAS software stack and the Athena framework, which must be made available in order to run jobs. Second, ATLAS production tasks are managed by PanDA and submitted to the WLCG, so a solution for submitting PanDA jobs to an HPC system jobs, monitoring them and getting back the results in an automated way is needed.

Enabling ATLAS Computing for HPC is a general goal within the ATLAS collaboration. The ATLAS HPC working group was founded to develop solutions for HPC integration. Within the HPC group, there are currently two complementary approaches for job submission and management on HPC sites: The ARC-based solution I developed, which has been tested at CSCS and at SuperMUC, and a lower level PanDA solution based on specialized HPC pilot jobs currently being tested at US HPC sites.

Possibilities for running the ATLAS software framework were evaluated on the CSCS „Todi" integration system in the first months of this work. Studies were focused on the two first steps in the ATLAS simulation workflow, the event generation and the detector simulation.

2.8 ATLAS software access on HPC

The ATLAS applications and workflows are different from usual self-contained HPC applications. In particular all the software and its dependencies are loaded and executed from the CVMFS [18] distributed file system. On WLCG clusters, CVMFS is mounted on the compute nodes using a kernel driver, which is not possible on HPC compute nodes. The ATLAS CVMFS repository root structure is lined out in Table 2.2. The repository contains data files, such as databases containing a description of the various parts of the detector, and different releases of the versioned ATLAS software stack. Each PanDA task specifies the ATLAS software release to use, which a Grid site needs to provide in order to run the job. Two possibilities were evaluated to achieve this on HPC systems; using „Parrot" virtual file system wrapper, and copying the software from CVMFS to a shared file system of the HPC facility.

Parrot software access Parrot is a virtual „file system wrapper" [20]. It starts a specified process, then uses the Linux process debugging interface („ptrace") to intercept system calls which access files, e.g. `open()`, `read()` or `write()`. This approach allows Parrot to simulate to the target process that certain files or file systems are available without actually mounting them into the file system. Parrot can run as a regular user process, while mounting the file system would require administrative (root) privileges.

Therefore, Parrot was evaluated as a possible solution for providing ATLAS jobs with access to CVMFS. Parrot was successfully built using the GCC compiler provided as `PrgEnv-gnu` on Cray machines, and with minor adjustments to the library search path, the ATLAS framework and associated software was successfully run for both event generation and detector response simulation jobs in single-threaded, single-node mode.

However, Parrot causes problems when running complex multi-threaded applications or hybrid CPU/GPU code. Running multi-threaded jobs using Parrot handling file access for all sub-processes showed multiple race-condition and deadlocking issues. While some of them were fixed in collaboration with the Parrot developers over the course of this work [21, 22, 23], larger-scale tests showed that multi-threaded jobs were still unstable, and new releases of the ATLAS software could trigger dormant bugs. Also, attempting to run GPU code or accessing GPU memory from an application running within Parrot crashes the application. Hence, Parrot, while being a very simple and clean solution in theory, was not used for production.

Directory	Contents	Required for production jobs
`ATLASLocalRootBase`	General, release-independent setup and software management scripts, e.g. for setting up specific versions of the various ATLAS software compontents. While users usually use this scripts to set up specific releases of the ATLAS software, production jobs set up the release directly and don't use ATLASLocalRootBase.	No
`conditions`	Symbolic link to the `atlas-condb` repository, which contains the ATLAS condition database, i.e. additional non-event data from the ATLAS detector [19]. It also holds detector parameters used for partially parametrized detector simulation (ATLAS fast simulation).	Yes
`dev`	Software development and testing area.	No
`sw/database`	Versioned ATLAS database, which contains e.g. the description of detector geometry and physical parameters. At least one (current) release of the database has to be provided in order to run production jobs.	Yes
`sw/atlas-gcc`	The GNU Compiler Collection (gcc) used to build the ATLAS software, including any associated libraries. Software dynamically linked need to access these libraries.	Yes
`sw/software`	Versioned ATLAS software stack. At the time of writing, the large-scale ATLAS Production tasks use the 17.7.3 and 17.7.4 releases of the ATLAS software for detector simulation and event generation, so at least these releases have to be provided in order to run current the considered steps of ATLAS production.	Yes

Table 2.2: The ATLAS CVMFS repository organization.

Part	Inode count
ATLAS software release (17.7.3)	427013
ATLAS condition database	8371
atlas-gcc	3062
Current ATLAS database release	1756
Total	**440202**

Table 2.3: Number of inodes (files and directories on the file system) used by the ATLAS CVMFS repository.

Maintaining a local CVMFS copy An alternative to using the Parrot wrapper is to keep a copy of the required ATLAS software on a shared file system at the HPC facility.

The first step is to copy the required parts of CVMFS (see Table 2.2) to a suitable file system, e.g. using the rsync utility from a system which has CVMFS mounted locally. It is to be noted that the ATLAS software consists of many small files, so while the total disk space needed less than 2 TB, the number of inodes[2] occupied is just below 450000 for a single release of the ATLAS software stack as shown in Table 2.3. This is much more than typical data of the same size would use. If the software is copied to backed up or versioned file system, this should therefore be agreed on with the HPC facility considering that the high inode count may increase the load on the versioning or differential backup system (e.g. CSCS enforces a limit of 500000 inodes on the versioned project storage).

After having copied the required parts from CVMFS, certain file system paths need to be patched, since the local CMVFS copy won't be accessible as `/cvmvs/` in the file system. While there are current efforts to make the ATLAS CVMFS repository fully relocatable, this is an ongoing task that has not been finished yet. Hence a script was developed within the ATLAS HPC working group to prefix every occurrence of `/cvmfs/` in the ATLAS setup scripts by a custom path.

For my tests on the „Todi" system, I copied the databases and two releases of the ATLAS software (17.7.3 and 17.7.4, which are currently being used for large-scale production tasks) to the local scratch file system, which is neither versioned nor backed up, and then used an improved version of the CVMFS relocation script to change absolute paths from `/cvmfs` to `/scratch/todi/mhoste/cvmfs`. I also implemented a complementary script to apply the same relocation technique to the ATLAS condition database, which was not covered by the original script.

[2] An inode represents a file or a directory on the file system

Full simulation time	~900 s/1 event
Memory usage	~2 GB
Job size	100 events
Input file size	< 300 MB/1000 events
Output file size	< 100 MB/100 events

Table 2.4: Typical ATLAS Geant4 full simulation job requirements

2.9 Choice of ATLAS Production step to run on HPC

Two steps of the ATLAS Monte Carlo production chain have been evaluated as possible workloads to run on HPC systems, the event generation step and the detector simulation step. In order to efficiently use HPC systems, the jobs run there should have high CPU time requirements, but only require low amounts of data to be copied in and out of the system, since the network connection of the HPC system is shared among all users. Both event generation and detector simulation jobs were run successfully in single-threaded mode on a single node of the „Todi" system, but currently PanDA only generates multi-core jobs for the detector simulation step. Also, detector simulation is the most CPU time consuming step of the whole simulation chain and uses up to 50% of the global Grid resources [16].

For this reasons, the detector simulation step was chosen for further studies. ATLAS detector simulation is based on the general-purpose particle interaction simulation toolkit Geant4 [24], which uses Monte Carlo methods to simulate the propagation of arbitrary particles through matter. In the ATLAS framework, Geant4 is compiled as a shared library, which is then used by the Athena framework. Athena is responsible for reading the detector geometry and parameters from the database files and setting up the Geant4 simulation geometry accordingly, and for providing the individual events read from an input data file to Geant4.

The typical requirements of a Geant4-based ATLAS detector simulation job are lined out in Table 2.4. It is to be noted that each input file contains 1000 events, while each job only processes a set of 100. However, when by using ARC and enabling the ARC caching feature, the input file can be reused by up to 10 subsequent jobs simulation. It can be concluded that data transfer, even if bandwidth is limited, poses no limitation to detector simulation jobs.

2.10 HPC Scaling and CPU performance optimization

After detector simulation was chosen as the ATLAS production workload to run on HPC, performance was assessed and optimized. It is important to note that on the Cray HPC systems targeted, the smallest resource a job can request from the resource manager is one compute node, i.e. nodes cannot be shared among jobs. This requires jobs to efficiently make use of all CPU cores available on a node. Also, possibilities for using GPUs of hybrid CPU/GPU HPC systems to speed up Geant4 simulations have been evaluated.

Multi-Threading

Multi-threading is required in order to use all the 16 CPU cores of the compute nodes efficiently, since 16 single-threaded jobs running in parallel could exhaust the available node memory (32 GB), while multi-threaded jobs can share large parts of the static data, e.g. the detector geometry. Parallelization is achieved at the event level (simulating 16 events in parallel) using AthenaMP, the multi-threading feature of the Athena framework. Since the detector simulation for each event is independent, no inter-process communication is needed apart from providing a shared queue of event numbers to simulate to the worker threads.

Profiling results, as depicted in Fig. 2.5, show an almost linear speedup when increasing the number of threads per job. The offset of ~30 min is due to the fixed time needed for initialization and finalization of the processing (e.g. loading the detector geometry or merging the results of the worker threads into a single file), which can only run serially on a single core. The memory usage of a 16-threaded job typically reaches ~6 GB in the end, which is a factor of 5 less than the 32 GB available, so memory usage is no concern when running multi-threaded jobs.

Scaling across multiple nodes

After using full compute nodes efficiently through multi-threading, possibilities for scaling across hundreds of nodes were assessed. Since all simulations are independent at the per-event level, there are two basic possibilities: Creating a few large multi-node jobs, then using inter-node communication trough the provided MPI system to distribute event numbers to the individual compute nodes; or creating many small, independent singe-node jobs, which process a number of events without any inter-node communication.

For creating multi-node jobs and distributing the events to process to individual nodes, AthenaMPI and the Yoda event service have been developed within the HPC working group. However, the problem of large jobs is that at the end of the list of events to

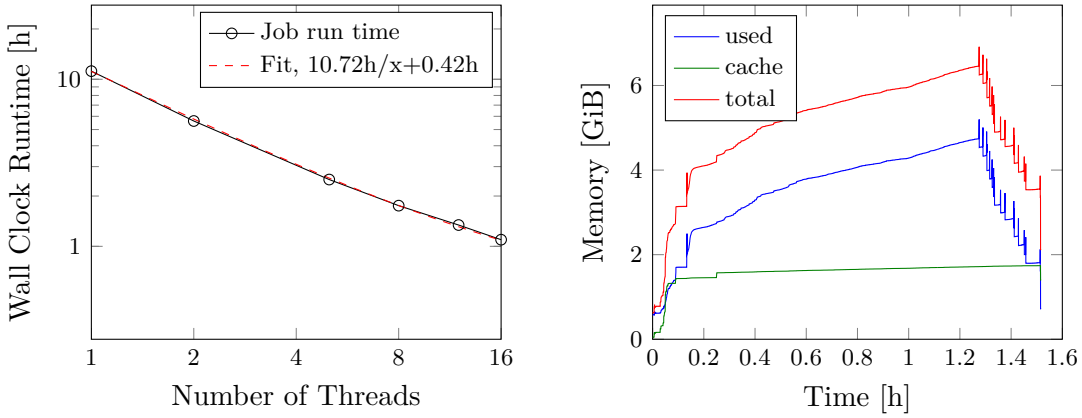


Figure 2.5: Thread-scaling of jobs. The scaling is near-perfect (linear), with a slight offset due to the initialization and finalization steps. The total memory usage of a 16-threaded job processing 100 events is much lower than the 32 GB available per node.

process, not all nodes will finish processing at the same time, since the processing time for a single event is not fixed, but follows a probabilistic distribution. In HPC scheduling, nodes reserved for a multi-node job will only be freed after the job finishes on all nodes, so computing time will be lost on the nodes which finish early.

When using single-node jobs, this problem does not occur, since the node can be freed as soon as it finished processing the assigned events. Since the Cray hybrid SLURM/ALPS resource manager can handle thousands of jobs efficiently with a marginal overhead [25], and since the number of jobs a single user can submit at CSCS is not limited and no minimum job size is enforced, I decided not to use multi-node jobs. Instead I run each ATLAS production job on a single node. The maximum number of simultaneous jobs to run can be configured in the HPC submission system; in a load test, scaling up to 100 nodes was tested, showing a linear increase in throughput (Table 2.5).

Requested simultaneous jobs	10	100
Average running jobs	10 ± 0	95.3 ± 4.5
Completion rate [jobs/h]	7.28 ± 3.04	68.8 ± 13.9

Table 2.5: Comparison of ATLAS simulation jobs running on 10 and 100 compute nodes in parallel.

CPU compiler optimization

The CSCS cray machines provide different compiling toolchains. I evaluated if recompiling the existing ATLAS Geant4 codebase using the Cray Compiler Collection (CrayCC) or the GNU Compiler (gcc) specifically optimizing for the compute node CPU architecture leaded to a significant speedup compared to the pre-compiled binaries from CVMFS. Both compilers were run using the settings recommended by Cray [26]. For comparison, a set of 10 events was simulated using the different Geant4 binaries on a single core on the „Todi" machine. Results for three different random seeds were compared in order to estimate the uncertainty introduced by different series of random numbers generated by the different compilers. The total processing time per event can be found in Table 2.6.

Random Seed	**Precompiled**	**Optimized gcc**	**CrayCC**
539155	880 s	834 s	1219 s
939155	879 s	833 s	1208 s
139155	887 s	840 s	1178 s

Table 2.6: Processing time per event for different ATLAS Geant4 builds.

The comparison indicates that a speedup of 5% is possible by replacing the pre-compiled Geant4 binaries from CVMFS by their counterparts generated by re-compiling the ATLAS Geant4 codebase on the HPC system using gcc with full optimization for the target architecture. The results do not encourage CrayCC as a subject of further studies. While the use of optimized gcc binaries is planned for a further production project on CSCS HPC systems [27], the recompiled binary would need a full validation before being used as part of the ATLAS production chain. Since the speedup is only 5%, the precompiled, validated binaries from CVMFS were used for further production tests done in this study.

2.11 GPU optimization studies

Since both the „Todi" integration system and the current CSCS flagship system „Piz Daint" are hybrid CPU/GPU systems featuring NVIDIA Tesla GPUs, possibilities for using the GPUs to accelerate certain parts of the simulations were studied. Due to the memory limitations of GPUs (6 GB of GPU memory for 2688 GPU cores), the event-level parallelization approach used for running Geant4 on multi-core CPUs was not an option. Even when using the latest version of Geant4, which includes explicit sharing of static data among the threads, processing an event in Geant4 typically uses ~40 MB per thread [28], which is a factor of 10 over the available GPU memory. Also, the Geant4 processing full events would result in very dissimilar code, where the GPU branching limitations would become a limiting factor.

There are ongoing studies on using GPU code in the Geant collaboration [29], but only prototypes exist at this stage, and it is yet to be studied if the GPU prototype code can and will be implemented into the Geant4 codebase, or if it will be part of the future Geant5 release, which ATLAS computing would have to adapt to and validate first.

As a feasible short-term optimization, for this study the performance gain when replacing the Geant4 random number generator (RNG) by a GPU counterpart was considered. For this purpose, a GPU-based RNG was implemented based on the cuRAND [30] library. The generator provides both flat and Gaussian distributions. To avoid a bottleneck when copying the generated random numbers from GPU to CPU memory, a double-buffered approach was chosen, where a front and a back buffer for both flat and Gaussian distributed random numbers is kept in the CPU memory. During the initialization, both buffers are filled. The front buffer is used to provide random numbers when the RNG is called. When the last random number is used, buffers are swapped, and GPU code is run in the background to refill the now empty back buffer, while the CPU can run other code (Fig. 2.6).

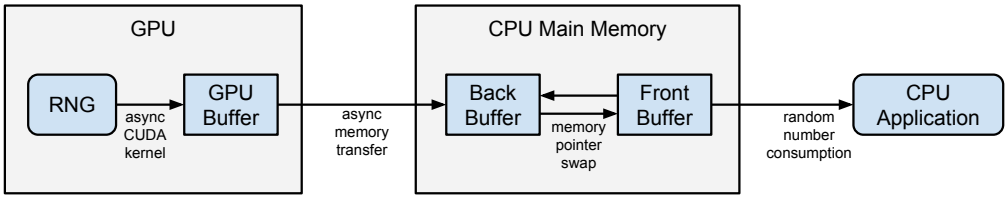


Figure 2.6: Principle of the double-buffered CUDA RNG

Tests have shown that generating flat distributed random numbers using the GPU-based RNG is a about a factor of 5 faster compared to the RNG previously used by Geant4, while the generation of Gaussian distributed random numbers is faster by a factor of 10. The ATLAS Geant4 codebase was then patched to use this CUDA-based RNG instead of the default Athena RNG. In total, a gain of 5% was observed by this improvement. While this is not much on itself, the study proved that dynamically including HPC specific optimizations in ATLAS Geant4 is possible.

2.12 Running and managing ATLAS Grid jobs on HPC

Apart from the basic possibility of running ATLAS software without administrative access or changing the system configuration, a system to automatically submit ATLAS jobs to the HPC system, monitoring them and eventually retrieving the results is a vital component for running ATLAS jobs on HPC machines. Running persistent services on HPC login nodes is usually not allowed, and access to the system for job submission and data management is only provided over the Secure Shell (SSH) protocol.

For job management and submission, the ARC middleware was extended by a back-end to run and manage jobs on a remote HPC system over SSH. This makes it possible to run an ARC-CE front-end to a HPC system on a server or virtual machine which is completely independent from the HPC system, and which does not even have to be in the same computing center. When using the arcControlTower for converting piloted PanDA jobs into regular ARC job, no internet access is required from the compute nodes of the HPC system. For this study, a virtual machine at the University of Bern was successfully used as a front-end to the „Todi" HPC machine at CSCS.

Following section 2.5, the ARC-CE middleware accesses the resource to manage (usually a local cluster, in my case a remote HPC system) and its Local Resource Management System (LRMS) in two ways:

- It prepares the session directory before submitting a job to the LRMS and retrieves the job output files from its session directory once a job completes. The job session directories must be shared between the ARC-CE host and the worker nodes running the jobs. This is commonly achieved by placing them on a shared file system.

- It communicates with the LRMS to submit jobs, to update the job and queue status in regular intervals, and to check if jobs are finished. This communication is done by Unix shell scripts parsing the output of the respective LRMS command line tools, e.g. `squeue` to check the status of the job queue if the local resource management system is SLURM.

To run an ARC-CE as a front-end to a remote HPC system therefore requires that both the access to the session directory and calls to the HPC scheduler are transparently handled over an SSH connection to the remote system. Also, scripts for interfacing with the LRMS may need to be adapted if the remote HPC system uses a non-standard LRMS. The integration concept is shown in Fig. 2.7, and is explained more in detail in the following.

File system access; SSHFS

SSHFS (SSH File System) is an open-source file system driver to mount and transparently access remote file systems over SSH and its SFTP (secure file transfer) functionality. In order to allow transparent access from the ARC-CE front-end to job session directories available on the HPC system, my approach mounts a shared file from the HPC system on the ARC-CE front-end through SSHFS. On the „Todi" system used for testing, the HPC „scratch" file system was chosen for this, since no long-term storage of session directories is required.

While SSHFS allows transparent access to the remote file system, care must be taken to synchronize the file system paths and file ownership between the ARC-CE front-end and the interfaced HPC system. ARC-CE expects that the path to the session directory is

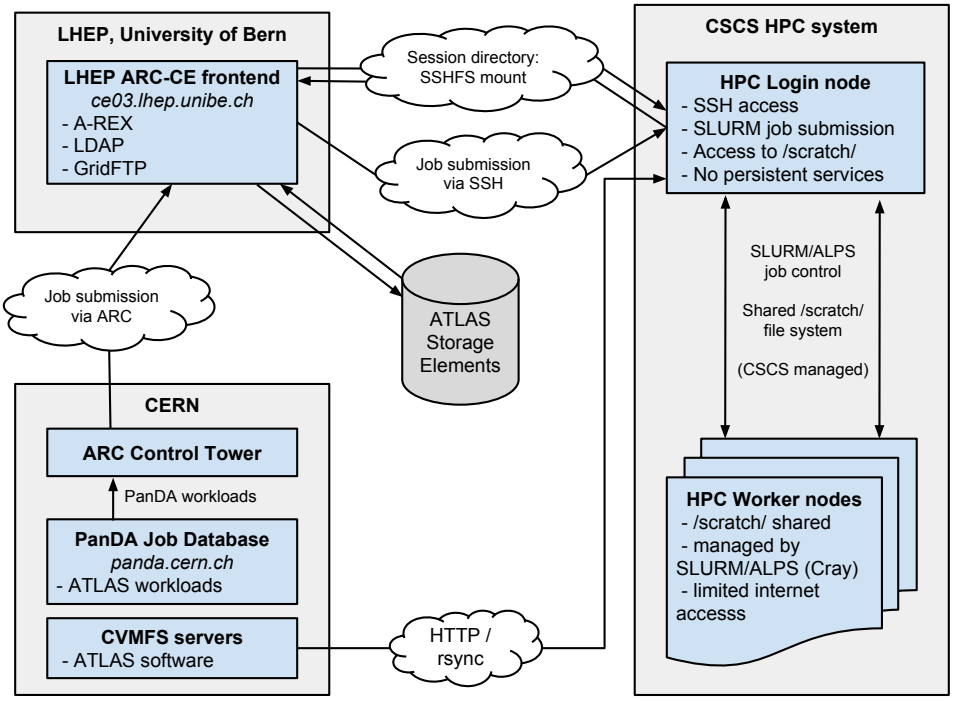


Figure 2.7: The architecture of the ARC-CE as a front-end to a remote HPC system. Clouds denote connections made over the internet.

the same on the front-end and on the worker node, e.g. `/scratch/todi/mhoste/` from the remote HPC system must be mounted to `/scratch/todi/mhoste/` on the ARC-CE front-end. To avoid file ownership problems, the user id and group id of the user account used to log into the remote HPC system must be mapped on the ARC-CE to both root and the local user ARC uses for running Grid jobs.

The remote file system mounted over SSHFS can also be used by ARC to maintain a cache of recently used input files on the remote HPC system. If a subsequent job requests the same input file, it does not have to be copied to the HPC system again, but can be directly reused from the cache. This requires the capability of creating hard links on the remote file system, which was introduced in OpenSSH 5.7 [31]. If the SSH server on the remote HPC system is older[3], a custom `sftp-server` binary can be specified, which is then called over the SSH connection. It is to be noted that SSHFS must be run in single-threaded mode when using the ARC-CE cache to avoid race conditions when creating and looking up cached files of parallel jobs.

A full example configuration for SSHFS, as it was used for the ARC-CE front-end to the „Todi" HPC system at CSCS, is shown in listings 2.1 to 2.3.

[3]Which was the case for the CSCS machines in question, e.g. „Todi" uses OpenSSH 5.1

Listing 2.1: SSHFS configuration used on the ARC-CE interface to „Todi". The remote file system to mount is `/scratch/todi/mhoste/`, residing on the scratch file system of „Todi".

```
sshfs mhoste@todi1.login.cscs.ch:/scratch/todi/mhoste/ /scratch/
   todi/mhoste/ \
  -o reconnect -o allow_other -o workaround=rename -o idmap=file \
  -o uidfile=/opt/sshslurm/config/sshfs-todi.uidmap \
  -o gidfile=/opt/sshslurm/config/sshfs-todi.gidmap \
  -o nomap=ignore -o sftp_server=/users/mhoste/openssh-6.6p1/sftp-
     server \
  -o ServerAliveInterval=30 -o ServerAliveCountMax=2 -s
```

Listing 2.2: SSHFS group ID map used for the ARC-CE interface to „Todi". The group ID of our account on the HPC system is 33100, the group used by ARC to run Grid jobs is „griduser-michi"

```
griduser-michi:31100
root:31100
```

Listing 2.3: SSHFS user ID map used for the ARC-CE interface to „Todi". The user ID of our account on the HPC system is 22651, the ARC user is „griduser-michi".

```
griduser-michi:22651
root:22651
```

Remote Resource Manager access

The CSCS Cray HPC systems in question use a Cray-specific combination of SLURM (Simple Linux Utility for Resource Management) and the Cray ALPS (Application Level Placement Scheduler) to manage jobs. Job scripts are submitted through the standard SLURM command line tool `sbatch` from the HPC login node, which queues the job and reserves the requested resources. When a job gets to run, SLURM executes the submitted job script on a HPC service node. The job script must then call the ALPS `aprun` command to execute a script or binary on the reserved compute nodes. The job status can be queried from the HPC login nodes through standard SLURM command line tools (e.g. `squeue` or `scontrol`).

To run and manage jobs on the HPC system through the ARC-CE front-end, I've implemented a script (`sshslurm`) to transparently call the SLURM command line tools on the remote system over SSH (listing 2.4). It uses the password-less SSH Public Key method for authentication and optionally allows using SSH connection sharing to minimize the resource overhead of the SSH connection. The script is not called directly, but by symbolic links named after the SLURM commands to run on the remote system (e.g. `sbatch`, `scancel`, `scontrol` or `squeue`). Note that for job submission through

`sbatch`, the job scripts needs to be copied to the destination system before using SCP. While my implementation is specific to SLURM, the principle is not, and the script can be easily adapted to access other HPC resource managers remotely.

Also, the ARC-CE interface to SLURM was modified to generate a special job script, which takes into account the hybrid SLURM/ALPS architecture of Cray HPC systems and runs the job through ALPS `aprun` when it is executed by SLURM.

Listing 2.4: sshslurm shell script to forward calls to SLURM job control tools over SSH.

```
#!/bin/bash

# config
source /opt/sshslurm/config/sshslurm-config

SBINARY=$(basename "$0")
SARGS=""
for token in "$@"; do
        SARGS="$SARGS '$token'"
done

if [[ "$SBINARY" == "sbatch" && "$1" != "" ]]; then
        SARGS=$REMOTE_TEMP_PATH/$(basename "$1")
        $SCP_CMDLINE -q "$1" "$SSHSLURM_HOST:$SARGS"
        $SSH_CMDLINE $SSHSLURM_HOST -- [ -d "$PWD" ] \&\& cd "$PWD
            "\; $REMOTE_SLURM_PATH/$SBINARY "$SARGS" \&\& rm -f "
            $SARGS"
        exit $?
fi

$SSH_CMDLINE $SSHSLURM_HOST -- [ -d "$PWD" ] \&\& cd "$PWD"\;
    $REMOTE_SLURM_PATH/$SBINARY "$SARGS"

exit $?
```

Listing 2.5: Configuration of sshslurm on the ARC-CE interface to „Todi". SSH connection sharing is used to minimize the time overhead.

```
SSHSLURM_HOST="mhoste@todi1.login.cscs.ch"
SSH_CMDLINE="/opt/openssh-6.6/bin/ssh -o "ControlPath=~/.ssh/
    controlmaster-%r@%h:%p" -o "ControlMaster=auto" -o "
    ControlPersist=2h" -o "ServerAliveInterval=120" -i /opt/
    sshslurm/config/id_rsa.$(whoami)"
SCP_CMDLINE="/opt/openssh-6.6/bin/scp -o "ControlPath=~/.ssh/
    controlmaster-%r@%h:%p" -o "ControlMaster=auto" -o "
    ControlPersist=2h" -o "ServerAliveInterval=120" -i /opt/
    sshslurm/config/id_rsa.$(whoami)"
REMOTE_SLURM_PATH="/opt/slurm/default/bin"
REMOTE_TEMP_PATH="/tmp"
```

2.13 Results

The HPC integration system developed was tested on the CSCS „Todi" Cray XK7 with regular ATLAS production jobs. Due to CSCS internal regulations, large-scale production tests were postponed to September 2014, when the machine ended its regular user operation. Tests included a short-term load test with 100 parallel ATLAS jobs on the machine, and a long-term stability tests with a maximum of 50 parallel jobs over several months. Both tests were completed successfully.

The ARC-CE front-end setup for „Todi" was completed by mid-September 2014. For about two weeks, it run the HammerCloud functional tests for multi-threaded ATLAS detector simulation [32] to confirm the basic functionality was working. The first ATLAS production jobs which delivered results were run September 26, with 10 jobs running in parallel (Fig. 2.8).

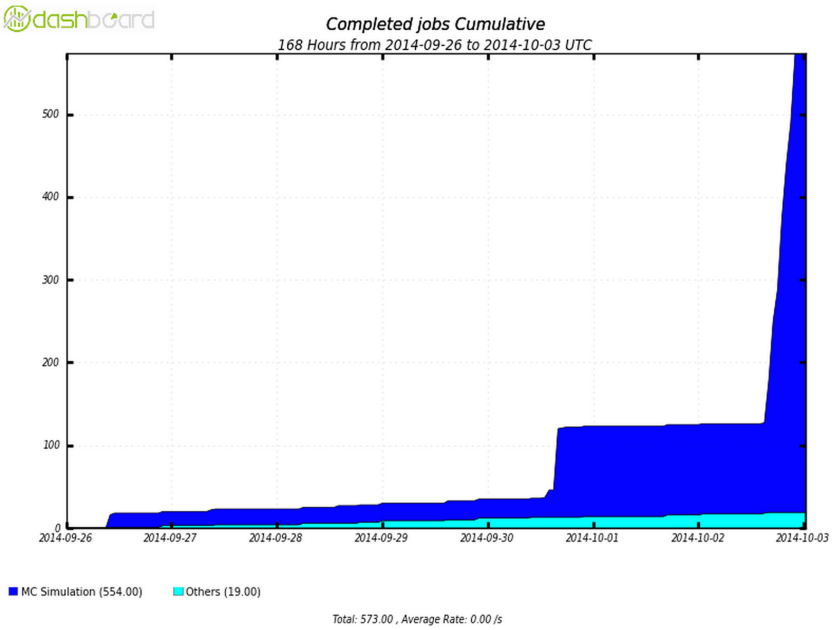


Figure 2.8: Screenshot of the ATLAS job monitoring system showing the first test runs on „Todi" using my HPC setup. First ATLAS production jobs were run starting September 26. On September 30, a first load test (50 parallel jobs) was started. The second load test (100 parallel jobs) started on October 2.

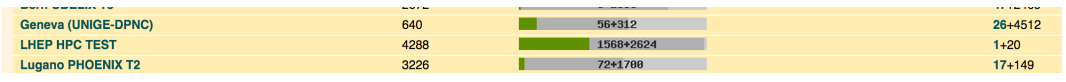


Figure 2.9: Screenshot of the Nordugrid ARC-CE monitor during the 100 job load test, showing a total of 1568 cores of the HPC system being used by ATLAS production jobs through my ARC-CE front-end (96 jobs using 16 cores each).

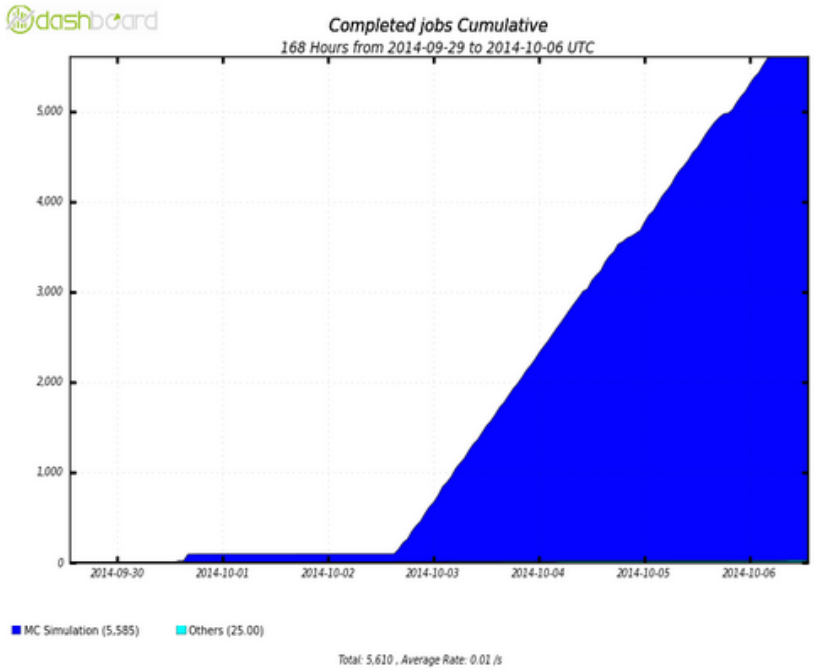


Figure 2.10: Screenshot of the ATLAS job monitoring system after the 100 job load test, showing more than 5000 successfully processed ATLAS production jobs.

After the setup was found to be stable, the behavior of the ARC-CE front-end under load was tested. For a first load test, 50 jobs were submitted to the ARC-CE in parallel on September 30. After the jobs completed successfully, a second test with 100 being submitted in parallel was done on October 2 (Fig. 2.9). The test was continued for five days with the arcControlTower continuously keeping 100 jobs running on the ARC-CE. This run was successfully completed with more than 5000 ATLAS production jobs (equivalent to ~150000 CPU-hours of computing time) processed and results uploaded to an ATLAS storage element (Fig. 2.10).

Once the load tests were complete, the ARC-CE front-end was reconfigured to run continuously up to 50 parallel production jobs on „Todi" to check the long-term stability of the integration solution. The limit to 50 parallel jobs was set to minimize the impact on the other users of the „Todi" system, since we did not apply for large-scale production in 2014 on CSCS HPC systems. Over the next months, the ARC-CE front-end was found to be stable, and contributed to ATLAS production at a level comparable to the ATLAS Tier-2 cluster run by the LHEP at the University of Bern. In total, ~26800 ATLAS production jobs equivalent to more than 500000 CPU-hours were processed on „Todi" by the end of January 2015 (Fig. 2.11, 2.12, 2.13). The limiting factor was the number of suitable multi-threaded production jobs available from PanDA.

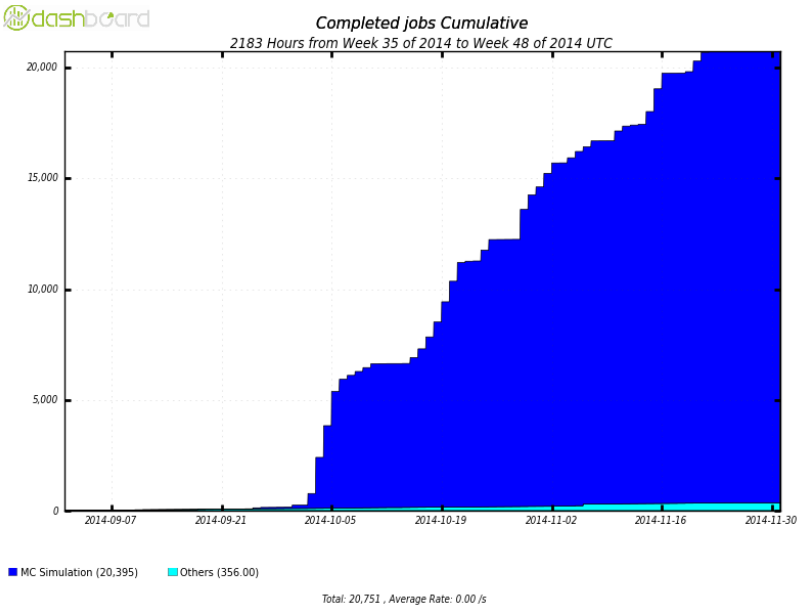


Figure 2.11: Screenshot of the ATLAS job monitoring system [8], showing the evolution of production jobs processed on „Todi" by the end of November 2014.

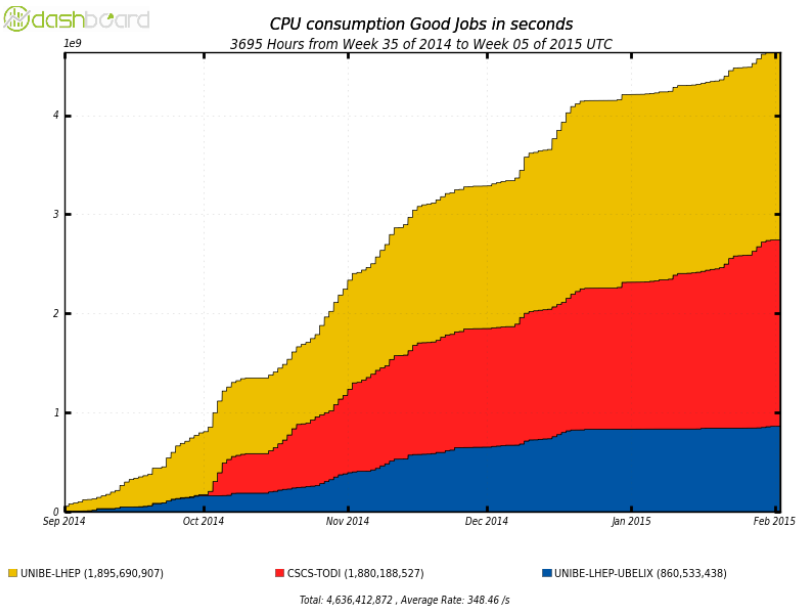


Figure 2.12: Stacked total CPU seconds contributed to ATLAS production, starting September 1, by the UNIBE-LHEP Tier 2 cluster (yellow), the UNIBE-UBELIX Tier 3 cluster (blue) and the „Todi" HPC system interfaced through the ARC-CE front-end (red).

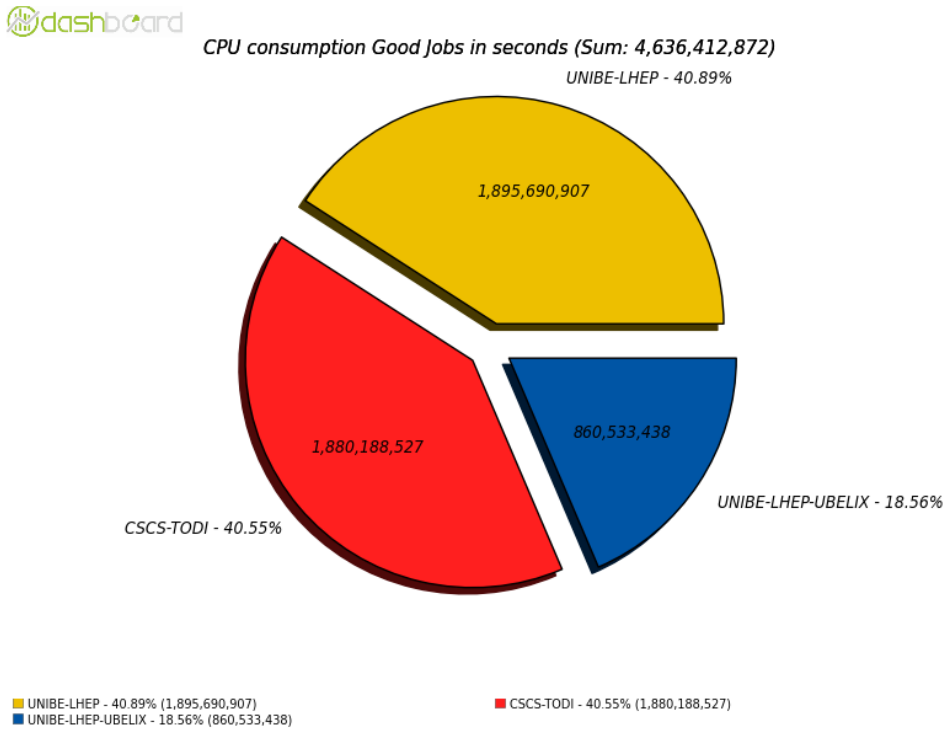


Figure 2.13: Comparison of the total CPU time share contributed to ATLAS production, as in Fig. 2.12. Contributions from UNIBE-LHEP and „Todi" are in the same range.

2.14 Conclusion, Outlook and Scalability

It can be concluded that this study showed the feasibility of running the detector simulation step of ATLAS production on HPC systems. With the ARC-CE based front-end and the ATLAS software and databases copied from CVMFS to a shared file system, ATLAS production jobs can be accepted from PanDA via arcControlTower and run on a HPC machine. The target queue to submit jobs to, as well as the maximum number of simultaneously running jobs on the HPC system, can be set in the ARC-CE configuration. This way, both using an allocated CPU time share and back-filling a HPC system in an opportunistic way by submitting jobs to a low-priority back-fill queue is possible.

In our tests, our HPC integration approach showed a stable throughput running 50 parallel jobs (using 16 CPU cores each, 800 CPU cores in total) over several months. Submission handling of up to 100 simultaneously submitted jobs worked in a short-term load test. If jobs are released staggered when starting up the ARC-CE front-end (e.g. 50 jobs every few minutes) to flatten the peak load on the ARC-CE, this implies that hundreds to thousands of jobs running in parallel on the HPC system could be managed.

The ARC-CE front-end we used, which interfaces the CSCS „Todi" system, currently runs on a dual-core virtual machine with 4 GB of RAM at the University of Bern. If the resources are not sufficient for managing the number of parallel jobs targeted on a larger HPC system in the future, this setup can be scaled up by moving the ARC-CE front-end to a physical server. If network throughput to CSCS turns out to be a limiting factor, it could also run on a server located in the CSCS facility. Even having multiple independent ARC-CE front-ends submitting ATLAS jobs to the same HPC system for increased throughput or reliability is possible.

An adapted version of this integration system, submitting jobs to a remote LoadLeveler resource manager instead of SLURM, is currently used by Rodney Walker on the SuperMUC HPC system at the Leibniz Supercomputing Centre in Munich, Germany. At the time of writing, SuperMUC and Todi are the only HPC systems that run unmodified ATLAS production jobs within the PanDA framework without installing a local WLCG front-end [33]. An ARC-CE front-end based approach will also be used to run ATLAS production on the Pi supercomputer in Shanghai, China, with first runs successfully completed by A. Filipcic, J.K. Nielssen and S. Raddum in January 2015. Plans exist to integrate other Chinese and European HPC systems. In the future, SSH LRMS access might become a common ARC-CE feature, which would allow all supported LRMS to be transparently used over SSH.

For further ATLAS production on CSCS HPC systems, my supervisor Sigve Haug has requested 50 million CPU-hours on the „Piz Daint" flagship HPC system for a production project starting in April 2015 [27]. The project will use the approaches described in this thesis. In the long term, production on CSCS HPC systems could replace the dedicated WLCG Tier-2 cluster of the Swiss Institute of Particle Physics (PHOENIX) at CSCS, as getting a share on a future HPC system might be more cost efficient than purchasing and running a separate cluster for the same throughput of jobs.

3 Supersymmetry discovery potential in the $1l2b$ channel

3.1 Motivation

The goal of this additional study was to check the discovery potential of supersymmetry (SUSY) in events with 1 lepton, 2 b-jets ($1l2b$) and missing transverse momentum in the final state, considering the expected integrated luminosity from the first year of data taking at $\sqrt{s} = 13\,\text{TeV}$. The $1l2b$ channel is studied at the University of Bern for electroweak production. From a set of randomly generated phenomenological minimal supersymmetric models (pMSSM), models giving high yields were identified. Those models were then checked against current exclusion limits obtained in LHC Run 1 at $\sqrt{s} = 8\,\text{TeV}$.

3.2 The standard model of particle physics

The Standard Model (SM) of particle physics is a quantum field gauge theory which combines the theories of quantum electro-dynamics (QED) and quantum chromo-dynamics (QCD) to describe all fundamental particles currently known and to explain three of the four fundamental forces, namely the electromagnetic force, the weak force and the strong force. While it does not take into account the weakest of the four forces, the gravity, its predictions are still in a very good agreement with current experimental data.

The particle content of the SM is shown in Fig. 3.1; note that in addition to the particles described in the following, the SM also contains the corresponding antiparticles[1]. The SM describes three generations of fermionic (half-integer spin) particles, which can be further divided into leptons and quarks. Each quark generation consists of a quark carrying a charge of 2/3 and one quark carrying a charge $-1/3$, while each lepton generation consists of a massive negatively charged lepton and a neutral, massless neutrino.

The SM also describes 5 types of bosonic (integer spin) particles. There are 4 types[2] of

[1] The antiparticle of a particle has the same quantum numbers, but its charge conjugated

[2] It is to be noted that the gluon appears in three colors, and there are positively and negatively charged W bosons, which are not distinguished when referring to „4 types of bosons".

gauge bosons for the three fundamental forces the SM describes: the gluon (g, strong force), the photon (γ, electromagnetic force), and the W and Z bosons (weak force). The last boson is the Higgs boson (H), which gives the particles mass through the Higgs mechanism.

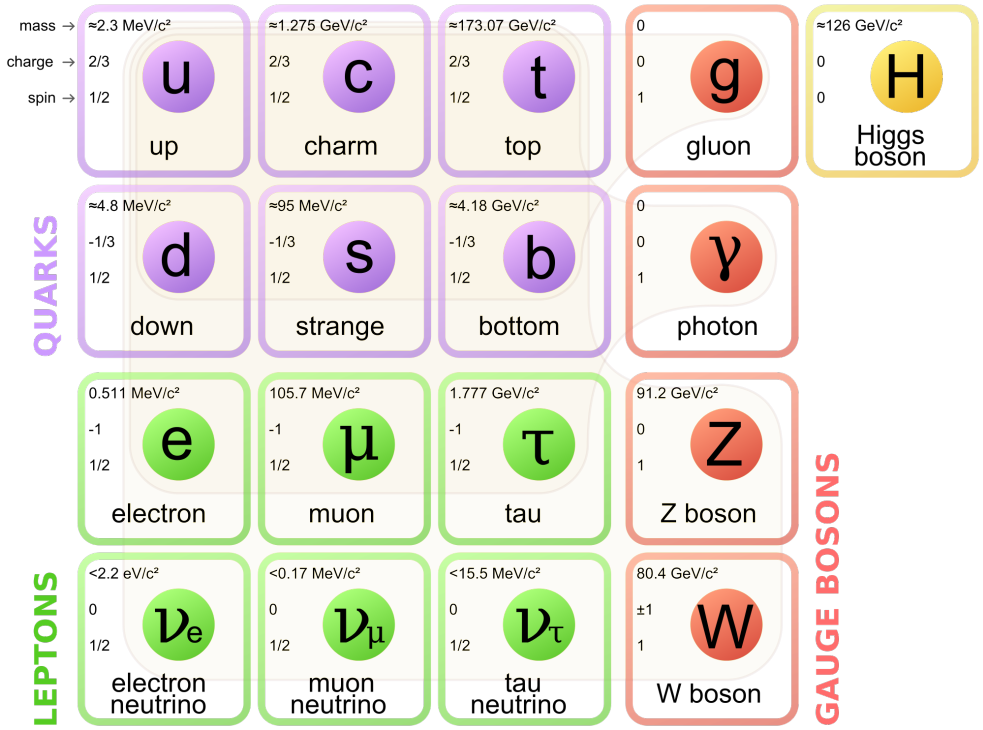


Figure 3.1: Particles described by the Standard Model. Note that the SM also describes the corresponding antiparticles, which are not shown here (taken from [34]).

The SM is based on three local gauge symmetry groups for the three fundamental forces described, $U(1) \times SU(2) \times SU(3)$. The $U(1)$ group is the symmetry group of the electromagnetic force, the $SU(2)$ group describes the weak force, and the $SU(3)$ group describes the strong force. To get a Lagrangian which is invariant under local gauge transforms in these symmetry groups, all gauge bosons and fermions must be intrinsically massless.

However, a significant mass is observed for the W and Z bosons, the leptons and the quarks. To solve this problem, the „Higgs mechanism" was proposed by P. Higgs, F. Englert and R. Brout in 1964 [35][36], based on the Goldstone theorem [37]. It allows fermions and gauge bosons to acquire mass by interacting with the Higgs field. This explains why a mass is measured for the fermionic particles of the SM, and also leads to the weak force being weak due to the high mass the W and Z bosons get. The Higgs field boson has been discovered by both the ATLAS and the CMS experiment in 2012 [38][39].

While the SM is in good agreement with current experimental data, there are some aspects which cannot be explained within SM physics, and therefore encourage theories beyond the SM:

- **Fine tuning and the hierarchy problem**: When calculating the Higgs boson mass, contributions in the order of Planck mass (10^{19} GeV) are expected from fermion loops. However, it is already known that the Higgs boson mass is 126 GeV.

  Assuming order of magnitude of the Planck mass is correct, the only way this problem could be solved within SM physics is to fine-tune SM parameters for the contributions to balance, which is not a natural solution. Theories beyond the SM can propose a fundamental way for the contributions to cancel out.

- **Dark Matter**: From cosmological observations, we know that ~27% of the total mass in todays universe must consist of Dark Matter, which only interacts weakly with observable matter. The SM does not provide any candidate particles for Dark Matter.

- **Massless Neutrinos**: The SM predicts the neutrino masses to be zero. From neutrino oscillation experiments, massless neutrinos are excluded.

- **Gravity**: The fourth fundamental force, the gravity, is not considered at all by the SM.

3.3 Supersymmetry

Super symmetric (SUSY) particle physics models assign each SM particle a „superpartner" (SUSY particle), which has the same quantum numbers, except that its spin differs by $\pm 1/2$. Hence, bosons get fermionic superpartners and vice versa. While introducing new particles, this can solve many of the SM issues noted above. In particular, contributions to the Higgs mass from SM quarks and their respective superpartner cancel out, which provides a natural solution to the hierarchy problem. Also those models can provide a natural Dark Matter candidate particle if the lightest SUSY particle (LSP) is stable and carries no charge.

However, pure supersymmetry predicts that SUSY particles have the same mass as their respective SM partner. This is excluded by experimental data, since this would lead to differences to the SM even at moderate energy scales, which would have been observed by past experiments. Different masses of SM particles and their superpartners can be explained by a broken supersymmetry.

The MSSM

The minimal supersymmetric model (MSSM) is a minimal extension of the SM obtained by making the following assumptions:

- **Minimal number of particles**: The MSSM introduces exactly one superpartner for each SM particle (Fig. 3.2). The bosonic superpartners of SM fermions are the squarks and sleptons, while the fermionic superpartners of SM bosons are gluino ($\tilde{g}$), higgsinos, wino and bino particles. Like the W_0 and the B boson mix in the SM, higgsino, wino and bino coupling eigenstates mix into chargino ($\tilde{\chi}^{\pm}_{1,2}$) and neutralino ($\tilde{\chi}^{0}_{1,2,3,4}$) mass eigenstates.

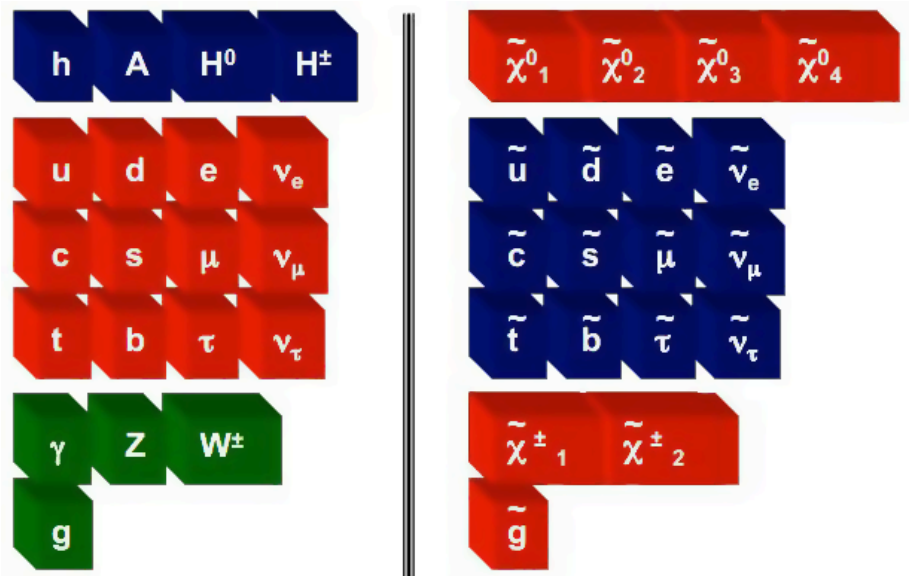


Figure 3.2: Particle spectrum of the MSSM. Note that the superpartners of the SM gauge and higgs bosons, the gauginos and the higginos, mix into neutralino and chargino mass eigenstates. (Based on [40])

  The MSSM also requires the introduction of a second Higgs field for the theory to be renormalizable. This leads to a total of 5 Higgs bosons (h, A, H^0, $H^{\pm}$) and their respective higgsino superpartners.

- **Soft SUSY breaking**: While it is known that the supersymmetry must be broken, the SUSY breaking mechanism is unknown. The MSSM assumes a minimal set of SUSY-breaking parametrized terms introduced to the Lagrangian [41]. The resulting theory is renormalizable without knowing the SUSY breaking mechanism in detail. However, the parametrized soft SUSY breaking terms introduce 105 free parameters to the MSSM.

- **R-Parity conservation**: The R-Parity of a particle is defined as

$$P_R = (-1)^{L+2s+3B}$$

where L is the lepton number, B is the baryon number and s is the spin. From this definition, it follows that $P_R = +1$ for SM particles and $P_R = -1$ for their SUSY superpartners.

The MSSM requires the R-Parity to be conserved. As a consequence, SUSY particles are always produced in pairs, and the lightest SUSY particle is stable. If the LSP is not charged (e.g. a neutralino), it is a natural dark matter candidate.

- **SM-like gauge symmetry group:** In the MSSM, the superpartners of the SM particles have the same $U(1) \times SU(2) \times SU(3)$ symmetry as their SM counterparts.

The full unconstrained MSSM introduces 105 additional free parameters. A parameter space with this many dimensions is too big to be fully analyzed. Hence constrained MSSM models, e.g. the pMSSM, make further assumptions in order to reduce the number of parameters.

The pMSSM

The so-called „phenomenological" MSSM (pMSSM) reduces the 105-dimensional parameter space of the MSSM to a set of 19 free parameters by introducing the following additional assumptions [42]:

- SUSY doesn't introduce any new source for CP-violation
- Minimal flavor violation, in particular, there are no flavor-changing neutral currents
- First and second generation squark universality

This leaves the 19 free parameters shown in Table 3.1. While a 19-dimensional parameter space is still big, regions can be excluded as the resulting pMSSM models would lead to physics, which is either excluded by theory or not consistent with present experimental data.

3.4 Decay Channels and Signal Regions

A particle detector like ATLAS cannot observe bare events and identify all primary particles involved for two reasons. First, many primary particles are not long-lived and decay before even passing through the detector and bare quarks hadronize, creating jets. Second, not all particles are detectable. In particular, particles which only interact

Parameter	Description
M_1, M_2, M_3	wino, bino and gluino mass parameters
A_u, A_d, A_e	first generation trilinear couplings (first, second and third generation are degenerate due to universality)
μ	Higgs-Higgsino mass parameter
$\tan(\beta)$	vacuum expectation value ratio of the neutral Higgs fields
M_A	pseudoscalar Higgs mass parameter
$M_{\tilde{e}L}$, $M_{\tilde{e}R}$, $M_{\tilde{q}1L}$, $M_{\tilde{u}R}$, $M_{\tilde{d}R}$	first generation sfermion masses (first and second generation are degenerate due to universality)
$M_{\tilde{\tau}L}$,$M_{\tilde{\tau}R}$, $M_{\tilde{q}3L}$, $M_{\tilde{t}R}$, $M_{\tilde{b}R}$	third generation sfermion masses

Table 3.1: pMSSM free parameters, following [42] and [43].

through the weak force (e.g. neutrinos or supersymmetric neutralinos) leave the detector undetected. Since the total transverse momentum is close to zero in head-on collisions, it is however possible to prove the existence of one or more invisible particles if detected particles miss transverse momentum.

Since supersymmetric particles are not expected to be directly detectable[3], one considers possible decay channels resulting in secondary particles which can be detected. If supersymmetric particles exists, they would create an excess in events over the expected standard model background. For this study, events with one lepton, two b-jets and missing transverse momentum in the final state were considered, which is usually referred to as the $1l2b$ channel. An example of an electroweak SUSY production mechanism contributing to the $1l2b$ channel is shown in Fig. 3.3.

To reduce the number of background events created by standard model processes and increase the signal-to-background ratio, further conditions (cuts) are applied against the detected particles. The region in parameter space included by a chosen set of conditions is called a signal region. The cuts applied for this study are described in detail in the next section.

[3]The lightest SUSY particle will not interact with the detector if it is not charged (therefore leaving missing transverse momentum), while the heavier SUSY particles are too short-lived to be detected.

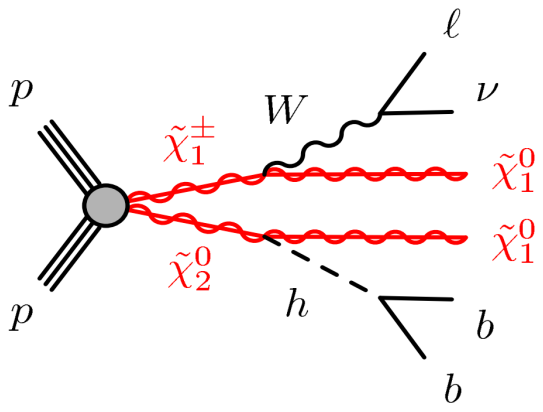


Figure 3.3: Chargino-neutralino based electoweak SUSY production in the $1l2b$ channel. Note that the neutrino and the two neutralinos leave the detector undetected, resulting in missing transverse momentum (taken from [44]).

3.5 Setup and Model Selection

For this study, the random valid pMSSM model set B. Geber generated [43] was used. Since this study was only a minor part of my thesis, only a subset of ~10% of the full 110500 valid models from B. Gerber was considered to reduce the processing time. As the models were generated by flat-random scanning the 19 dimensional pMSSM parameter space in no particular order, the picked models still cover the full parameter space of the original set (see Table 3.2 for parameter ranges), although providing less statistics. The models were already checked against basic exclusion conditions by B. Gerber during the generation process (listed in Table 3.3).

This study required a lot of events to be generated and run through a detector simulation quickly. For event generation, I used *Herwig++* [45], a Monte Carlo event generation

Parameter	**Range**
$tan(\beta)$	$[10, 60]$
μ	$[-2000, 2000]$
$A_{u,d,e}$	$[-10000, 10000]$
M_A	$[150, 3000]$ GeV
$M_{1,2,3}$	$[50, 3000]$ GeV
$m_{\tilde{e}_R,\tilde{e}_L,\tilde{\tau}_R,\tilde{\tau}_L}$	$[50, 3000]$ GeV
$m_{\tilde{q}_1,\tilde{q}_3}$	$[50, 3000]$ GeV
$m_{\tilde{u}_R,\tilde{t}_R,\tilde{d}_R,\tilde{b}_R}$	$[50, 3000]$ GeV

Table 3.2: Parameter ranges of the pMSSM models considered (taken from [43]).

E V O L U T I O N O F T H E M O D E L G E N E R A T I O N:

- Total nr. of generated flat-random-19-parameter-tupels: **18'685'986**

  EXCLUDED MODELS DUE TO SUSPECT2:
  - > Nr. of total SuSpect2-excluded models because of sfermionic tachyons, non-convergent μ, RGE problems etc.: 18'103'097

- Nr. of generated complete models: **582'889**

  EXCLUDED MODELS DUE TO SUPERISO-RELIC:
  - > Nr. of total superiso-relic-excluded models: 344'421
    - – Nr. of ERROR 1, excluded Higgs mass: 0
    - – Nr. of ERROR 2, excluded SUSY mass: 51'844
    - – Nr. of ERROR 3, charged LSP: 308'661

- Nr. of accepted models after SuperIso Relic: **238'468**

  EXCLUDED MODELS DUE TO branching ratios:
  - > Nr. of total BR excluded models: 127'968
    - – Nr. of $b \to s + \gamma$ excluded BRs: 111'308
    - – Nr. of $B_s \to \mu + \mu$ excluded BRs: 43'176
    - – Nr. of $B_d \to \mu + \mu$ excluded BRs: 11'857
    - – Nr. of $B \to \tau + \nu$ excluded BRs: 19'135

- Nr. of FINAL valid models after BR-filtering: **110'500**

Table 3.3: Output of B. Gerbers code to generate pMSSM models, mentioning the basic exclusion conditions checked during the model generation (taken from [43]).

package. For detector simulation, using the Geant4-based Monte Carlo approach as in first part of this study was not an option due to its processing time requirements, so I used the *DELPHES*-based [46] fully parametric ATLAS detector simulation. Both steps were run on the local computing cluster of the LHEP at the University of Bern.

The analysis was split into electroweak, strong and third generation SUSY production channels to compare their yields (Table 3.4). First, for each model considered, 15000 events for each channel were generated at $\sqrt{s} = 13\,\text{TeV}$. Detector simulation was run on the events, and events satisfying the $1l2b$ cuts set were counted. Models with more than

Production Channel	Outgoing Particles
Electroweak	$\tilde{\chi}^0_{1,2,3,4}$, $\tilde{\chi}^\pm_{1,2}$, $\tilde{\ell}$, $H^\pm$
Strong	$\tilde{u}$, $\tilde{d}$, $\tilde{c}$, $\tilde{s}$, $\tilde{g}$
Third generation	$\tilde{t}$, $\tilde{b}$

Table 3.4: Different production channels considered.

3 events[4] expected in an integrated luminosity of $30\,\text{fb}^{-1}$ were considered potentially sensitive in the $1l2b$ channel, and the event generation, detector simulation and event counting steps were repeated at $\sqrt{s} = 8\,\text{TeV}$. The yields were then checked against the model-independent exclusion limits of 5.6 events in $20.3\,\text{fb}^{-1}$ (at 95% confidence level) published in [44].

Cut Variable	**8 TeV Exclusion**	**13 TeV Sensitivity Check**
Number of b-jets	Exactly 2	Exactly 2
Jet kinematics	b-jets must be leading jets, $p_T > 20\,\text{GeV}$	b-jets must be leading jets, $p_T > 20\,\text{GeV}$
Jet Veto	No fourth-leading jet with $p_T > 25\,\text{GeV}$	none
Number of leptons	Exactly one e/μ	Exactly one e/μ
Lepton kinematics	$E_\ell > 25\,\text{GeV}$	$E_\ell > 25\,\text{GeV}$
E_T^{miss} Missing transverse momentum	$> 100\,\text{GeV}$	$> 100\,\text{GeV}$
m_{CT} Boost-corrected contransverse mass [47]	$> 160\,\text{GeV}$	$> 160\,\text{GeV}$
m_T Transverse mass of lepton and E_T^{miss} [44]	SRA: $100 < m_T < 130\,\text{GeV}$ SRB: $m_T > 130\,\text{GeV}$	$> 130\,\text{GeV}$
m_{bb} b-jet invariant mass [44]	$105 < m_{bb} < 135\,\text{GeV}$	none

Table 3.5: $1l2b$ cuts applied for 8 TeV exclusion and 13 TeV sensitivity check simulations.

[4] 3 events would be the sensitivity limit for a discovery in a hypothetic background-free scenario

Table 3.5 lists the $1l2b$ cuts applied for the event counting. The cuts applied for the sensitivity check at $\sqrt{s} = 13\,\text{TeV}$ were discussed with F. Meloni of the Bern ATLAS Team for this work, while the cuts for the exclusion at 8 TeV are consistent with [44] for comparability.

3.6 Results

From the set of 110500 random pMSSM models B. Gerber generated, 10000 models were tested. Testing the $1l2b$ sensitivity at $\sqrt{s} = 13\,\text{TeV}$ failed for 598 models due to Herwig++ not being able to generate events for particular models. Therefore, only the remaining 9402 models were considered for this study.

Models with $1l2b$ sensitivity

Histograms of the amount of models yielding to more than 3 events in $30\,\text{fb}^{-1}$ of $\sqrt{s} = 13\,\text{TeV}$ data in the $1l2b$ channel for strong, electroweak and third generation production are shown in Figs. 3.4-3.6. About 13% of the models were found to have potential $1l2b$ sensitivity on strong and ~10% on third generation production, while only ~1% of the models showed electroweak production sensitivity. Some models were found to be sensitive to multiple production channels. Statistics are aggregated in Table 3.6.

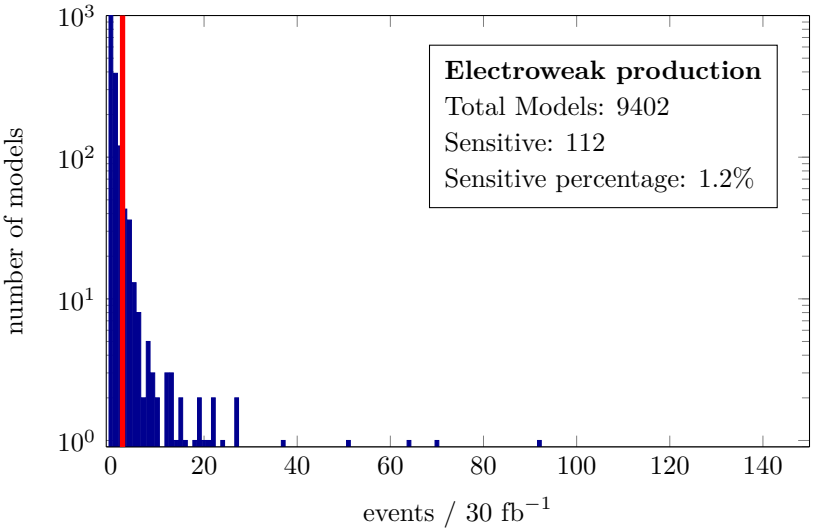


Figure 3.4: Histogram of the number of pMSSM models producing $1l2b$ events at $\sqrt{s} = 13\,\text{TeV}$ by electroweak SUSY production. The red line marks the assumed sensitivity limit of 3 events per $30\,\text{fb}^{-1}$

Production channel	$1l2b$ **sensitivity**	**Production channels**	$1l2b$ **sensitivity in both channels**
Electroweak	113 (1.20%)	Electroweak and Strong	10 (0.11%)
Strong	1260 (13.40%)	Electroweak and Third Generation	7 (0.07%)
Third generation	926 (9.85%)	Strong and Third Generation	147 (1.56%)
Total models with $1l2b$ sensitivity in any channel: 2042 (21.72%)			

Table 3.6: Sensitivity of electroweak, strong and third generation SUSY production channels for $1l2b$ events at $\sqrt{s} = 13\,\text{TeV}$ for the 9402 models tested.

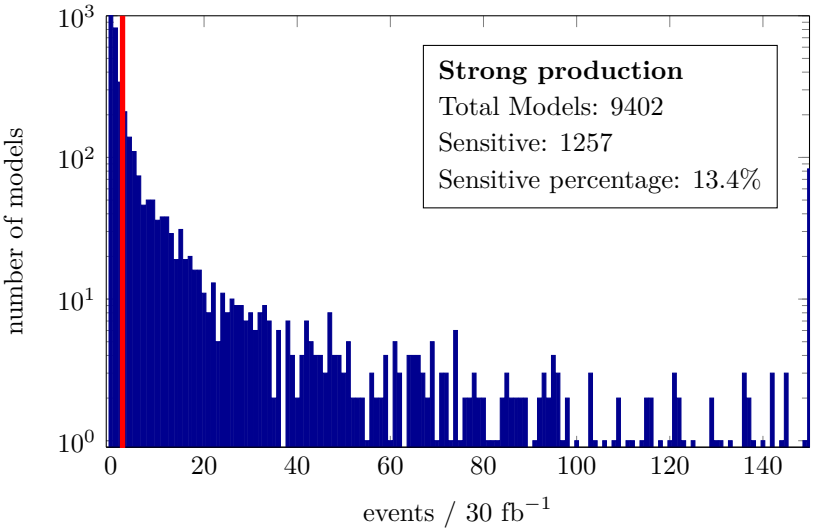


Figure 3.5: Histogram of the number of pMSSM models producing $1l2b$ events at $\sqrt{s} = 13\,\text{TeV}$ by strong SUSY production. The red line marks the assumed sensitivity limit of 3 events per $30\,\text{fb}^{-1}$

Excluded Models

For the 2042 models where the $1l2b$ channel was found to be potentially sensitive at $\sqrt{s} = 13\,\text{TeV}$ were run through a second simulation chain at $\sqrt{s} = 8\,\text{TeV}$ to check event yields against known exclusion limits. Less than 1% of the potentially sensitive models was found to be excluded, and the second simulation chain failed for ~10% of the sensitive models. Hence ~90% of the sensitive models were confirmed to be not excluded by the $\sqrt{s} = 8\,\text{TeV}$ data considered. The final statistics are presented in tables 3.7 and 3.8.

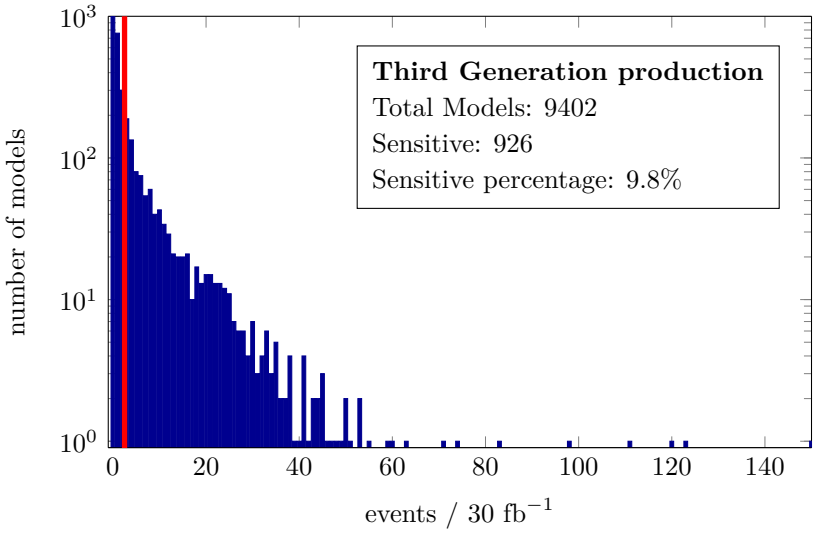


Figure 3.6: Histogram of the number of pMSSM models producing $1l2b$ events at $\sqrt{s} = 13\,\text{TeV}$ by third generation SUSY production. The red line marks the assumed sensitivity limit of 3 events per $30\,\text{fb}^{-1}$

Exclusion statistics	
Excluded models	7 (0.34% sensitive, 0.07% total)
Failed models	208 (10.19% sensitive, 2.21% total)
Non-excluded models	1820 (89.13% sensitive, 19.36% total)

Table 3.7: Statistics on models with $1l2b$ sensitivity excluded by $\sqrt{s} = 8\,\text{TeV}$ data.

Mass spectra and pMSSM parameters of sensitive, non-excluded models

For the 1820 models which were both found to be potentially sensitive and confirmed to be non-excluded, the mass spectra and the distribution pMSSM parameters have been analyzed. The resulting histograms, stacked by the production channel the respective model is sensitive to, are shown in Figs. 3.7 and 3.8. It is to be noted that there are models with $1l2b$ sensitivity at the far end of the model set's parameter limits (e.g. 3000 GeV for the mass parameters), which could become a limitation when reusing this pMSSM model set for simulations at higher energies.

In the mass spectrum, the negative masses for the $\tilde{\chi}_0^2$ in many models comes from the fact that the neutralinos are mixed states [48]. Note that the $1l2b$ channel tends to be more sensitive to third generation production for models with a $\tilde{g}$ mass above ~1500 GeV, while for models with lower $\tilde{g}$ masses it is more sensitive to strong production. Also, third generation production is strongly favored if $\tilde{t}_1$ and $\tilde{b}_1$ masses are below ~1000 GeV, which

Non-excluded models sensitivity	
Electroweak production	105 (1.12%)
Strong production	1121 (11.92%)
Third Generation production	822 (8.74%)

Table 3.8: Number of non-excluded models with $1l2b$ sensitivity by production channel.

could be explained by a larger phase space volume and hence a higher third-generation branching ratio if third-generation squarks are light.

The pMSSM parameter distribution can be compared with the distribution of the full model set B. Gerber generated [43]. Considering the distributions, a possible reference pMSSM model to have a high $1l2b$ sensitivity through third generation production should have a negative A_d of about -0.5, a positive A_u of ~0.3, a M_2 below 100 GeV and a M_3 higher than 1500 GeV. Also, M_{bR} and M_{q3L} should be below 1000 GeV. For a model sensitive to $1l2b$ through strong production, limits are much broader, as the parameters of models with high $1l2b$ sensitivity are more evenly distributed. For electroweak production, no conclusion is possible due to the low statistics of $1l2b$ events through electroweak production.

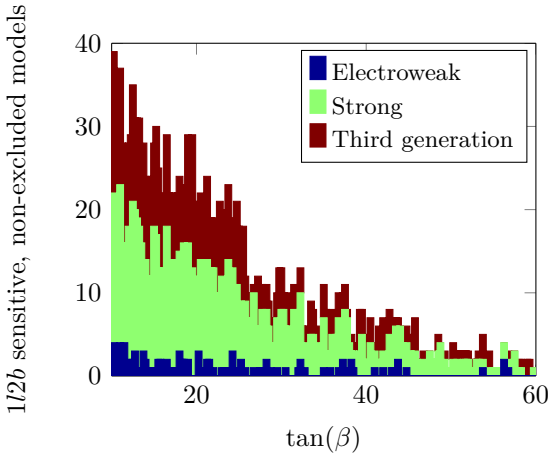


Figure 3.7: Stacked histograms of the pMSSM parameters of the non-excluded models with $1l2b$ sensitivity by production channel.

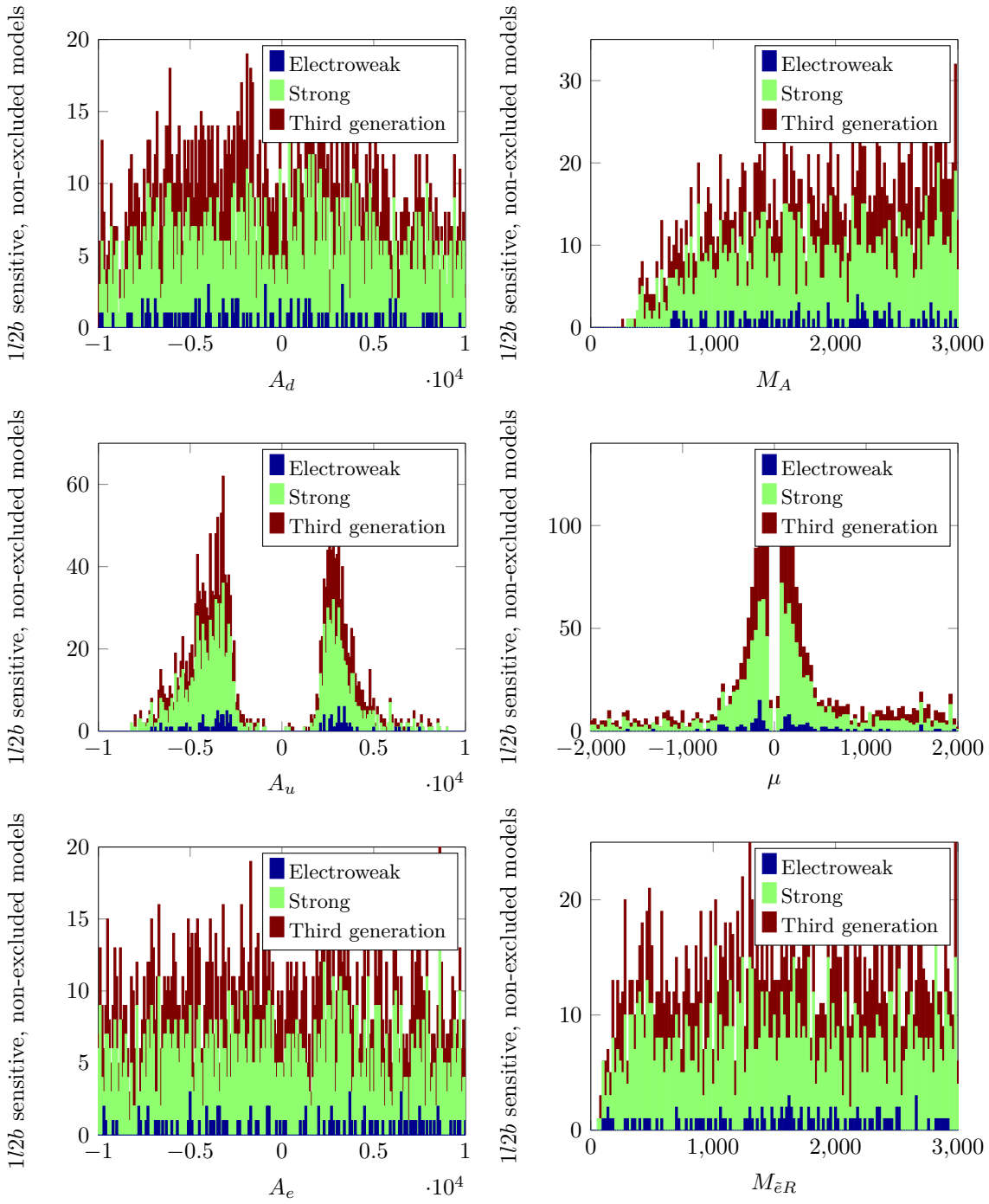


Figure 3.7 continued: Stacked histograms of the pMSSM parameters of the non-excluded models with $1l2b$ sensitivity by production channel.

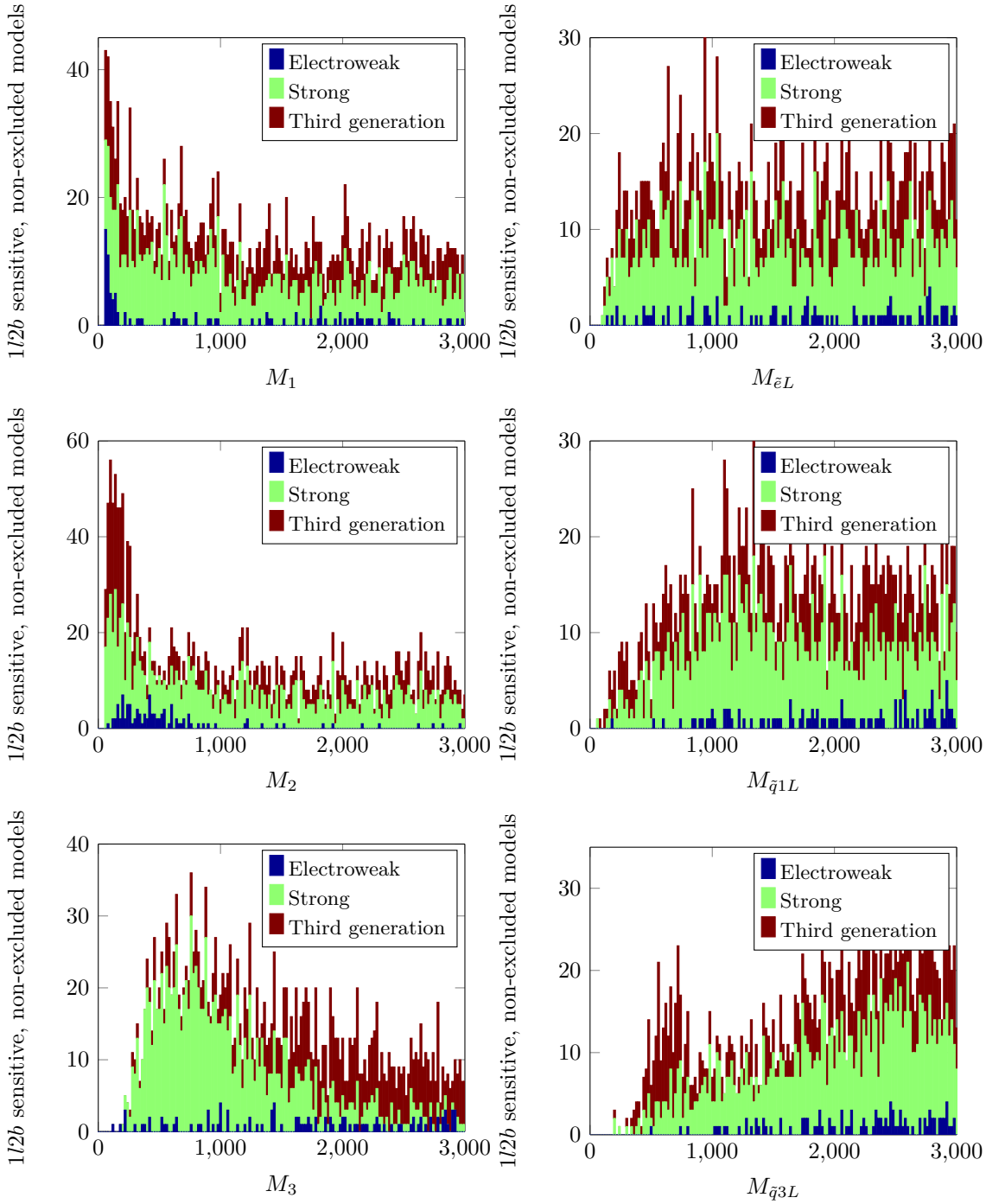


Figure 3.7 continued: Stacked histograms of the pMSSM parameters of the non-excluded models with $1l2b$ sensitivity by production channel.

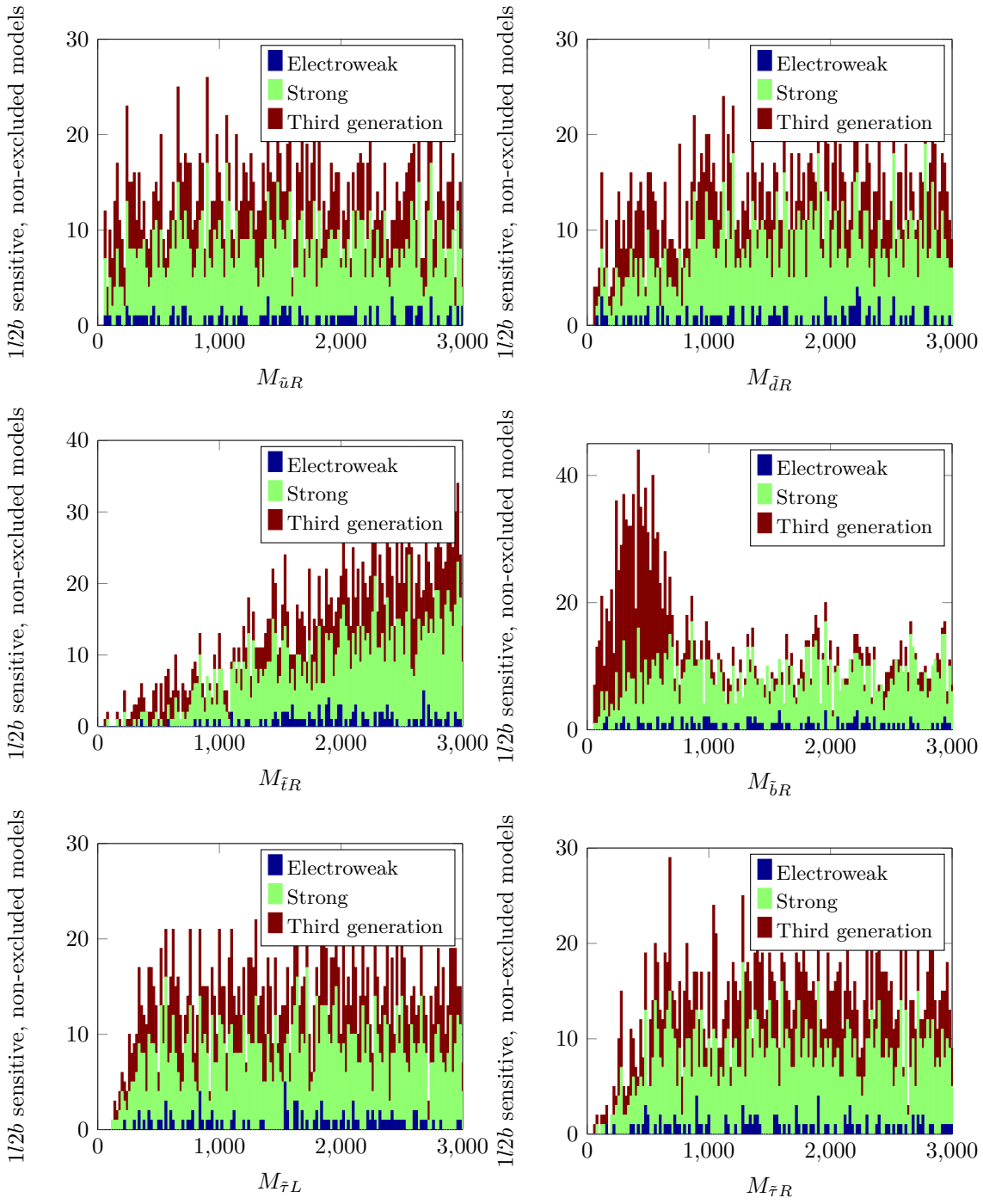


Figure 3.7 continued: Stacked histograms of the pMSSM parameters of the non-excluded models with $1l2b$ sensitivity by production channel.

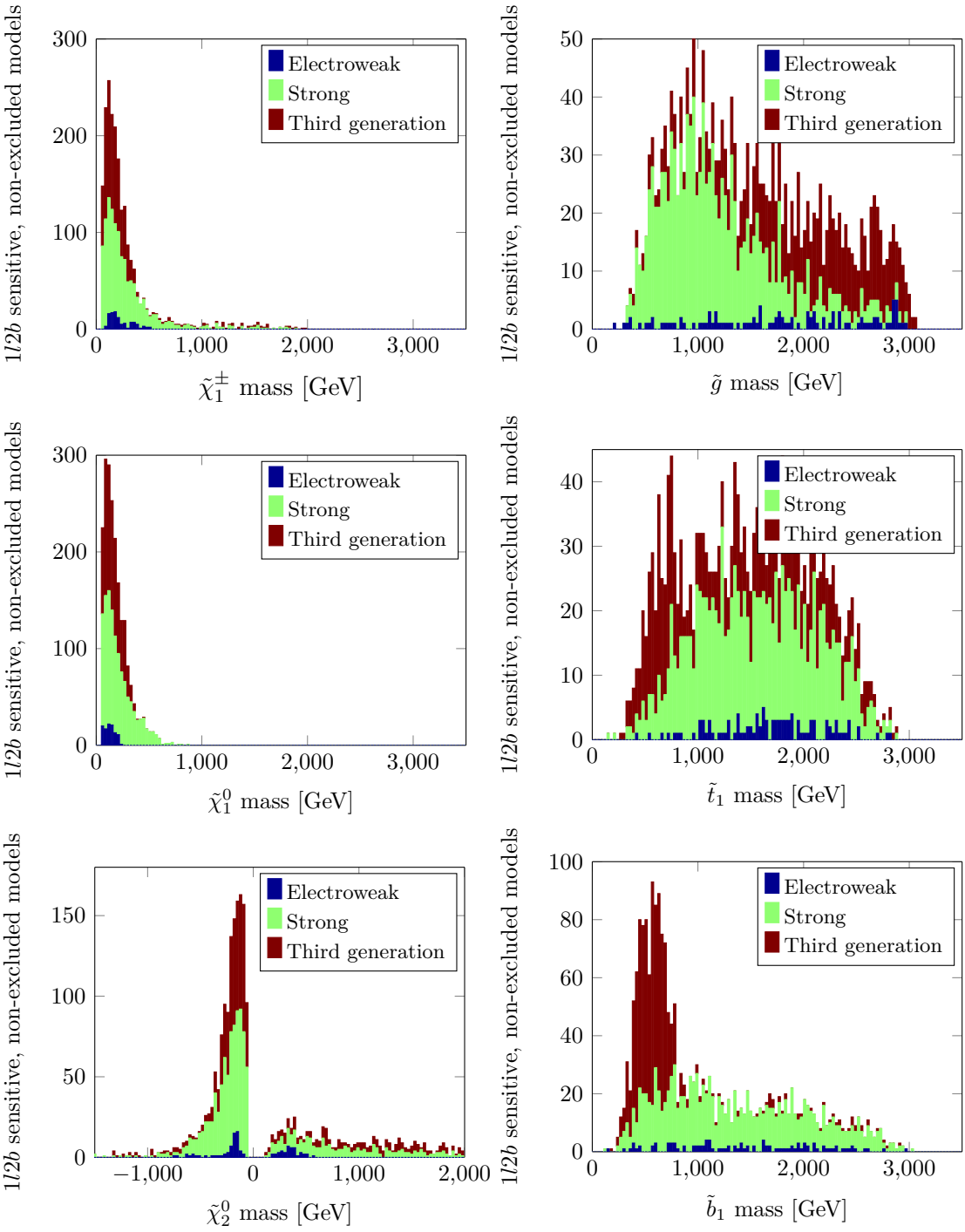


Figure 3.8: Stacked histograms of the mass spectra of the non-excluded models with 1ℓ2b sensitivity by production channel.

3.7 Conclusion

The study showed that the $1l2b$ decay channel is potentially sensitive to ~22% of the models considered at $\sqrt{s} = 13\,\text{TeV}$ when assuming a background-free measurement, where the discovery limit would be 3 events in $30\,\text{fb}^{-1}$ of integrated luminosity. Of those ~22%, the strong SUSY production channel is sensitive to the most models (~13% of total), third generation production is second (~10%), while electroweak production only gives $1l2b$ sensitivity for ~1% of the models.

The models with potential $1l2b$ sensitivity were checked against the model-independent exclusion limits for the 1l2b channel from ATLAS data taken at $\sqrt{s} = 8\,\text{TeV}$ [44]. It was shown that less than 10% of the models with $1l2b$ sensitivity at $\sqrt{s} = 13\,\text{TeV}$ are excluded, so ~20% of the total models considered have $1l2b$ sensitivity at $\sqrt{s} = 13\,\text{TeV}$ and are not excluded by the experimental data considered.

From the mass spectra, it can be concluded that third generation production is favored over strong production for events with $1l2b$ final states if the $\tilde{g}$ mass is high and $\tilde{t}$ and $\tilde{b}$ masses are low. The full pMSSM parameter histograms indicated a limited optimal region for reference models to have $1l2b$ sensitivity through third generation production, while the parameter regions for $1l2b$ sensitivity through strong production are way broader.

4 Summary

This Master thesis was driven by the goal to enable the search for new physics with the ATLAS detector at the LHC accelerator at CERN. In the first part, a solution for running ATLAS Monte Carlo simulation programs on general-purpose High-Performance Computing (HPC) systems using the ARC Grid middleware was developed, which has the potential to significantly increase the computing power available to the ATLAS experiment in a cost-effective manner, therefore providing the necessary statistics in the search for new physics phenomena.

It was shown that the solution works on the „Todi" HPC integration system at the Swiss National Supercomputing Centre (CSCS). For this test, real ATLAS production jobs were used and more than 500000 CPU-hours of computing power were provided to the ATLAS collaboration over ~6 months, which is roughly equivalent to the share of the dedicated ATLAS clusters used by the LHEP group at the University of Bern. The same solution runs on the „SuperMUC" system in Germany, and a similar approach is currently being tested to use the Chinese „Pi" HPC system for ATLAS production jobs.

Furthermore, I evaluated possibilities for optimizing the current ATLAS detector simulation codebase for particular HPC systems. The performance gain from recompiling the detector simulation software Geant4 with a specific CPU-architecture compiler optimization showed a speedup at the ~5% level. Also, feasibility of replacing parts of the Geant4 code with equivalent GPU code within the ATLAS software framework was demonstrated. As a test, the Geant4 random number generation was implemented on the GPU, leading to a ~5% performance gain.

The second part of the thesis work was focused on the search for new physics by the analysis of ATLAS events, namely on determining the discovery potential for Supersymmetry (SUSY) in events with one lepton, two b-jets and missing transverse momentum in the final state. This channel ($1l2b$) is the subject of specific studies being conducted by the ATLAS group of LHEP at the University of Bern. My work on the analysis of this specific SUSY channel provided valuable information for targeting future searches, notably those following the forthcoming LHC run 2 at higher beam energy and luminosity.

In conclusion, I found that the $1l2b$ channel is only sensitive to electro-weak SUSY production for ~1% of the pMSSM models considered in the study. Depending on the mass of the third-generation supersymmetic squarks, either strong (sensitive to ~12% of the pMSSM models) or third-generation production (sensitive to ~9% of the models) is predominant for the $1l2b$ sensitivity.

5 Acknowledgements

At this point, I would like to thank the University of Bern, CERN, the ATLAS Collaboration and the Swiss National Supercomputing Centre (CSCS) for providing the resources to make this thesis possible, and certain people for their great support.

At the University of Bern, I thank Prof. Dr. Antonio Ereditato for general supervision, helpful comments on my thesis, and for leading the Laboratory for High Energy Physics to make such work possible at all. For suggesting the topic, answering my questions, proofreading, and very fruitful discussions on both computing and physics, thanks to my supervisors PD Dr. Sigve Haug and Prof. Dr. Michele Weber. For special computing and Grid expertise and support when setting up and running my Grid front-end, thanks to Dr. Gianfranco Sciacca. For very helpful discussions on SUSY and the $1l2b$ channel, thanks to Dr. Federico Meloni. Also, many suggestions from and discussions with all members of the Bern ATLAS team are warmly acknowledged.

At CERN, within the ATLAS collaboration, thanks to the HPC working group for possibilities to present and discuss this work and getting constructive feedback. In particular, I would like to thank Prof. Dr. Andrej Filipcic, Dr. Rodney Walker and Dr. David Cameron for great discussions on possibilities to integrate HPC systems into the Grid, development of the arcControlTower, feedback on the ARC-CE front-end approach, and testing of this approach on foreign HPC systems.

At CSCS, thanks to Pablo Fernandez and Miguel Gila for the support and the possibility to test the automated ARC-CE submission on the „Todi" HPC system.

Bibliography

[1] The ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, J. Instrum. 3 (2008) S08003 and CERN, Switzerland, 2008.

[2] The CMS Collaboration, *The CMS Experiment at the CERN LHC*. JINST 3 (2008) S08004 and CERN, Switzerland, 2008.

[3] The LHCb Collaboration, *The LHCb Detector at the LHC*. JINST 3 (2008) S08005 and CERN, Switzerland, 2008.

[4] The ALICE Collaboration, *The ALICE Experiment at the CERN LHC*. JINST 3 (2008) S08002 and CERN, Switzerland, 2008.

[5] K. Schindl, *The injector chain for the LHC*. 9th LEP-SPS Performance Workshop, France, 1999.

[6] The ATLAS Collaboration, *ATLAS Computing: Technical Design Report*, CERN, Switzerland, 2005.

[7] I. Bird et al., *Update of the Computing Models of the WLCG and the LHC Experiments*, CERN, Switzerland, 2014.

[8] The ATLAS Job Dashboard, `http://dashb-atlas-job.cern.ch/dashboard/request.py/dailysummary`.

[9] Simple Linux Utility for Resource Management (SLURM), `https://computing.llnl.gov/linux/slurm/`.

[10] Open Grid Scheduler/Grid Engine, `http://gridscheduler.sourceforge.net/`.

[11] NVIDIA, *CUDA Toolkit Documentation*
`http://docs.nvidia.com/cuda/parallel-thread-execution/#set-of-simt-multiprocessors-with-on-chip-shared-memory`

[12] ATLAS@Home, `http://atlasathome.cern.ch/`

[13] D. Cameron et al., *The Advanced Resource Connector for Distributed LHC Computing*, `http://atlasathome.cern.ch/`, XII Advanced Computing and Analysis Techniques in Physics Research, Italy, 2008.

[14] The Nordugrid Collaboration, *Extended Resource Specification Language*, 2014. http://www.nordugrid.org/documents/xrsl.pdf

[15] The ATLAS Nordugrid Monitor, http://www.nordugrid.org/monitor/atlas/.

[16] The ATLAS Collaboration, *Concepts and Plans towards fast large scale Monte Carlo production for the ATLAS Experiment*, ATLAS Note, CERN, Switzerland, 2013.

[17] A. Filipcic, *arcControlTower: the System for Atlas Production and Analysis on ARC*, J. Phys. Conf. Ser. **331** 072013, 2011.

[18] The ATLAS Collaboration, *Software installation and condition data distribution via CernVM FileSystem in ATLAS*, ATLAS Note, CERN, Switzerland, 2012.

[19] The ATLAS Collaboration, *ATLAS Computing Technical Design Report*, CERN/LHCC/2005-022, CERN, Switzerland, 2005.

[20] The Parrot Virtual File System, http://ccl.cse.nd.edu/software/parrot/.

[21] Parrot crashes when forked process reading from boost shared memory, Parrot bug report, https://github.com/cooperative-computing-lab/cctools/issues/360

[22] Parrot unnecessarily keeps mmap()'ed files open, Parrot bug report, https://github.com/cooperative-computing-lab/cctools/issues/419

[23] Parrot: terminated threads reported by waitpid(), Parrot bug report, https://github.com/cooperative-computing-lab/cctools/issues/431

[24] The Geant4 Collaboration, *Geant4, a simulation toolkit*, Nuclear Instruments and Methods in Physics Research A 506 (2003) 250-303, 2003.

[25] A. Heart (Cray), Private Communication, CSCS Piz Daint course, Lugano, Switzerland, 2014.

[26] A. Lazzaro (Cray), *Overview of Compilers on XC30*, CSCS Piz Daint course, Lugano, Switzerland, 2014.

[27] S. Haug et al., *Enabling LHC searches for the unknown content of the universe*, CSCS CHRONOS Project Proposal for 2015 allocation, Switzerland, 2014.

[28] The Geant4 Collaboration, *Geant4, towards major release 10*, CHEP 2013, Amsterdam, Netherlands, 2013.

[29] The Geant4 Collaboration, *GPUs in Geant4*, Annual Concurrency Meeting, Fermilab, USA, 2013.

[30] NVIDIA, *cuRAND*, http://docs.nvidia.com/cuda/curand/.

[31] Add attribute extensions to sftp-server, OpenSSH feature request, `https://bugzilla.mindrot.org/show_bug.cgi?id=1555`.

[32] F. Legger, *Improving ATLAS grid site reliability with functional tests using HammerCloud*, ATL-SOFT-PROC-2012-007, ATL-COM-SOFT-2012-019, CERN, Switzerland, 2012.

[33] A. Klimentov, *High Performance Computing (i.e. supercomputing) in ATLAS*, ATLAS weekly presentation, CERN, Switzerland, 2015.

[34] Wikipedia article on the Standard Model, `http://en.wikipedia.org/wiki/Standard_Model`.

[35] F. Englert, R. Brout, *Broken symmetry and the mass of gauge vector mesons*, Physical Review Letters, 13(9):321-323, 1964.

[36] P. W. Higgs, *Broken Symmetries and the Masses of Gauge Bosons*, Physical Review Letters, 13:508-509, 1964.

[37] J. Goldstone, A. Salam, S. Weinberg, *Broken Symmetries*, Physical Review 127: 965-97, 1962.

[38] The ATLAS Collaboration, *Observation of an Excess of Events in the Search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, CERN, Switzerland, 2012.

[39] The CMS Collaboration, *Observation of a new boson with a mass near 125 GeV*, CERN, Switzerland, 2012.

[40] C. Young, *ATLAS SUSY Multi-Jet Search*, BOOST 2013 Conference Slides, USA, 2013.

[41] K. Müller, *Soft SUSY Breaking*, Lecture Notes, Universität Zürich, Switzerland, 2002.

[42] A. Djouadi et al., *The Minimal supersymmetric standard model*, Group summary report, 1998.

[43] B. Gerber, *A systematic approach to the search for SUSY in the 19-dimensional pMSSM parameter space in the ATLAS experiment*, Master Thesis, University of Bern, Switzerland, 2014.

[44] B. Butler et al., *Search for Chargino and Neutralino Production in Final States with One Lepton, Two b-jets Consistent with a Higgs Boson, and Missing Transverse Momentum with the ATLAS detector in* $20.3\,\mathrm{fb}^{-1}$ *of* $\sqrt{s} = 8\,\mathrm{TeV}$ *pp collisions*, ATLAS Note, CERN, 2014.

[45] M. Bahr et al., *Herwig++ Physics and Manual*, Eur. Phys. J. C58:639-707, 2008.

[46] S. Ovyn et al., *Delphes, a framework for fast simulation of a generic collider experiment*, arXiv:0903.2225, 2010.

[47] D. R. Tovey, *On measuring the masses of pair-produced semi-invisibly decaying particles at hadron colliders*, JHEP0804:034, 2008.

[48] V. A. Beylin, *Diagonalization of the neutralino mass matrix and boson-neutralino interaction*, Eur. Phys. J. C56:395-405, 2008.

Erklärung

gemäss Art. 28 Abs. 2 RSL 05

Name/Vorname: Hostettler Michael

Matrikelnummer: 08-112-856

Studiengang: Master of Physics

Bachelor ☐ Master ☐ Dissertation ☐

Titel der Arbeit: Enabling the ATLAS Experiment at the LHC for High Performance Computing

LeiterIn der Arbeit: Prof. Antonio Ereditato

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe r des Gesetzes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.

Ich gewähre hiermit Einsicht in diese Arbeit.

.....

Ort/Datum

.....

Unterschrift