From collisions to analysis

Vladimir V. Gligorov  CNRS/LPNHE
CERN-FNAL Summer School 09-2017
What will you learn in these lectures?

How is data collected & processed prior to analysis?
What will you learn in these lectures?

How is data collected & processed prior to analysis?

What types of processings exist, how do we choose between them and how do they affect later analysis?
What will you learn in these lectures?

How is data collected and processed prior to analysis?

What types of processings exist, how do we choose between them and how do they affect later analysis?

How do we calibrate and understand this processing?
What will you learn in these lectures?

How is data collected and processed prior to analysis?

What types of processings exist, how do we choose between them and how do they affect later analysis?

How do we calibrate and understand this processing?

In short: what I wish I’d been taught starting out!
Why do we need to process data (in real-time)?
How much data do our detectors record?

Scale: CMS is $O(100M)$ electronics channels, LHCb is $O(1M)$
How much data does LHCb process?

Input data rate of the LHCb experiment today ~ 1 TB/second
How much data does LHCb process?

Input data rate of the LHCb experiment today ~ 1 TB/second

This means about 4000 PB of data every year.
Which is quite some “real world” data

Input data rate of the LHCb experiment today ~ 1 TB/second

This means about 4000 PB of data every year

Google was at ~7000 PB/year in 2008, so goodness knows where it is today...

AT&T networks

BBC iPlayer

Facebook

Twitter

NB : ATLAS/CMS about a bit more than one order of magnitude above LHCb
This is a lot more than we usually quote.
This is a lot more than we usually quote

The volume of data produced at the Large Hadron Collider (LHC) presents a considerable processing challenge.

Particles collide at high energies inside CERN's detectors, creating new particles that decay in complex ways as they move through layers of subdetectors. The subdetectors register each particle's passage and microprocessors convert the particles' paths and energies into electrical signals, combining the information to create a digital summary of the "collision event". The raw data per event is around one million bytes (1 Mb), produced at a rate of about 600 million events per second.

The data flow from all four experiments for Run 2 is anticipated to be about 25 GB/s (gigabyte per second)

• ALICE: 4 GB/s (Pb-Pb running)
• ATLAS: 800 MB/s – 1 GB/s
• CMS: 600 MB/s
• LHCb: 750 MB/s
This is a lot more than we usually quote.

The volume of data produced at the Large Hadron Collider (LHC) presents a considerable processing challenge. Particles collide at high energies inside CERN’s detectors, creating new particles that decay in complex ways as they move through layers of subdetectors. The subdetectors register each particle’s passage and microprocessors convert the particles’ paths and energies into electrical signals, combining the information to create a digital summary of the "collision event". The raw data per event is around one million bytes (1 Mb), produced at a rate of about 600 million events per second.

The data flow from all four experiments for Run 2 is anticipated to be about 25 GB/s (gigabyte per second):

- ALICE: 4 GB/s (Pb-Pb running)
- ATLAS: 800 MB/s – 1 GB/s
- CMS: 600 MB/s
- LHCb: 750 MB/s

That’s because the data has already been processed in real-time!
Basic real-time processing

A zero-suppression algorithm for the readout electronics of the SciFi Tracker for the LHCb detector upgrade

H. Chanal
Published 24 February 2016 • © 2016 IOP Publishing Ltd and Sissa Medialab srl
Journal of Instrumentation, Volume 11, February 2016
Topical Workshop on Electronics for Particle Physics

Abstract

A new detector made of scintillating fibres read out by arrays of silicon photomultipliers (SiPM) is planned for the LHCb detector upgrade, foreseen in 2018/19. The development of dedicated readout electronics in the harsh LHC environment leads to challenges. Each SiPM array generates 10.24 Gb/s of data after the digitization leading to a data rate of 47.2 Tb/s for the full detector. Such a large amount of data can not be reasonably processed by a computing farm. In this paper, we describe the readout scheme and the zero suppression algorithm used to reduce the data flow below 8 Tb/s.

Most basic real-time processing is zero-suppression. Gain x5-10 in the size of the data, lossless from POV of physics information
Less basic real-time processing

Real-time data selection ("trigger") reduces more but lossy
Data processing is either lossless or lossy for the physics you are trying to measure.

If the processing is lossy you have to account for its impact on your measurement.

Different types of processing can be lossy for some physics analyses and lossless for others.
But let’s come back to our detector

Imagine you really needed to use all the detector raw data for your analysis. Could you physically do it?
Could you read all the data out?

1 MHz (Triggered) - planned:
- Network:
  - 1 MHz with ~5 MB: aggregate ~40 Tbps
  - Links: Event Builder-cDAQ: ~ 500 links of 100 Gbps
  - Switch: almost possible today, for 2022 no problem
- HLT computing:
  - General purpose computing: 10(rate)x3(PU)x1.5(energy)x200kHS6 (CMS)
    - Factor ~50 wrt today maybe for ~same costs
    - Specialized computing (GPU or else): Possible

40 MHz (Triggerless) – not planned:
- Network:
  - 40 MHz with ~5 MB: aggregate ~2000 Tbps
  - Event Builder Links: ~2,500 links of 400 Gbps
  - Switch: has to grow by factor ~25 in < 10 years, difficult

Front End Electronics
- Readout Cables: Copper Tracker! – Show Stopper

HLT computing:
- General purpose computing: 400(rate) x3(PU)x1.5(energy)x200kHS6 (CMS)
  - Factor ~2000 wrt today, but too pessimistic since events easier to reject w/o L1
  - This factor looks impossible with realistic budget
- Specialized computing (GPU or …)
  - Could possibly provide this …

With thanks to Wesley Smith

Not a high-luminosity hermetic detector, but otherwise…
It depends on your detector geometry

...you can actually read the full zero-suppressed detector out without making the material budget of your detector infinite
And it depends on your data rate

The same is true if you have a hermetic detector but you have a lower data rate (or can compress it), for example in ALICE.

<table>
<thead>
<tr>
<th>Detector</th>
<th>Input to Online System (GByte/s)</th>
<th>Peak Output to Local Data Storage (GByte/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC</td>
<td>1000</td>
<td>50.0</td>
</tr>
<tr>
<td>TRD</td>
<td>81.5</td>
<td>10.0</td>
</tr>
<tr>
<td>ITS</td>
<td>40</td>
<td>10.0</td>
</tr>
<tr>
<td>Others</td>
<td>25</td>
<td>12.5</td>
</tr>
<tr>
<td>Total</td>
<td>1146.5</td>
<td>82.5</td>
</tr>
</tbody>
</table>
Data processing also a physics choice...

Analysis aims to observe particles & force-carriers produced in LHC collisions, measure their production rates, and measure the rate at which they decay into other particles & force-carriers.
Most “interesting” particles&force-carriers decay before leaving a signal in the detector, so we have to measure their properties by reconstructing and studying their decay products. These products are typically known SM particles and can be thought of as common building blocks for all analyses.
What is this common reconstruction?

**Raw data from detector**
What is this common reconstruction?

Raw data from detector

Common processing

Charged particle trajectories  Neutral clusters  Jets  Composite objects
What is this common reconstruction?

Raw data from detector

Common processing

Charged particle trajectories  Neutral clusters  Jets  Composite objects

If you have particle-ID

π/K/p/e/μ
What is this common reconstruction?

Raw data from detector

Common processing

Charged particle trajectories
Neutral clusters
Jets
Composite objects

If you have particle-ID

If you have particle-ID

Using CALO, tracker

\(\pi/K/p/e/\mu\)

\(\gamma/e/\pi^0/K^0/\ldots\)
What is this common reconstruction?

Raw data from detector → Common processing → Charged particle trajectories → Neutral clusters → Jets → Composite objects

If you have particle-ID → \(\pi/K/p/e/\mu\)

Using CALO, tracker → \(\gamma/e/\pi^0/K^0/\ldots\)

Combine charged/neutral → \(T/D^0/MET/\ldots\)
Imagine if each analysis did all of this

Almost every analysis has two basic components:

1. Combine & select building blocks to observe a signal above background
2. Understand the efficiency of step (1) to measure signal properties
Imagine if each analysis did all of this

If each group of analysts made their own “common” objects, how would you do this? How would you combine the results of the different analyses? How would you determine correlations between their systematics?
Nota bene, this is not purely hypothetical

Digging Deeper for New Physics in the LHC Data

Pouya Asadi, Matthew R. Buckley, Anthony DiFranzo, Angelo Monteux and David Shih

*NHEC, Dept. of Physics and Astronomy
Rutgers, The State University of NJ
Piscataway, NJ 08854 USA*

*arXiv:1707.05783v1*

In this paper, we describe a novel, model-independent technique of “rectangular aggregations” for mining the LHC data for hints of new physics. A typical (CMS) search now has hundreds of signal regions, which can obscure potentially interesting anomalies. Applying our technique to the two CMS jets+MET SUSY searches, we identify a set of previously overlooked $\sim 3\sigma$ excesses. Among these, four excesses survive tests of inter- and intra-search compatibility, and two are especially interesting: they are largely overlapping between the jets+MET searches and are characterized by low jet multiplicity, zero $b$-jets, and low MET and $H_T$. We find that resonant color-triplet production decaying to a quark plus an invisible particle provides an excellent fit to these two excesses and all other data – including the ATLAS jets+MET search, which actually sees a correlated excess. We discuss the additional constraints coming from dijet resonance searches, monojet searches and pair production. Based on these results, we believe the wide-spread view that the LHC data contains no interesting excesses is greatly exaggerated.

Increasing focus on looking for beyond SM effects by combining the results of different individually inconclusive searches. A well calibrated common processing of the data is crucial for this.
Recap: we have to process data because raw data recorded by detectors too big to store.
Recap: we have to process data because

- Raw data recorded by detectors too big to store
- Most analyses use same SM building blocks, allows analysts to not reinvent all wheels for every analysis
Recap: we have to process data because

- Raw data recorded by detectors too big to store
- Most analyses use same SM building blocks, allows analysts to not reinvent all wheels for every analysis
- A consistent processing of SM objects recorded by detector helps when combining different analyses
Recap: we have to process data because

- Raw data recorded by detectors too big to store
- Most analyses use same SM building blocks, allows analysts to not reinvent all wheels for every analysis
- A consistent processing of SM objects recorded by detector helps when combining different analyses
- Now let's see how quick this processing should be
How real is time? Fixed latency vs. cascades of disk buffers
Every data processing step has to buffer the data which is being processed, before sending a (reduced) data volume to the next processing step. Data rate and buffer size determine the speed.
Types of buffers which are available in detector electronics
Types of buffers which are available

In detector electronics

In server farm near the detector ("software trigger")
Types of buffers which are available

- In detector electronics
- In server farm near the detector ("software trigger")
- In server farm far from the detector ("GRID")
Types of buffers which are available

- In detector electronics
- In server farm near the detector ("software trigger")
- In server farm far from the detector ("GRID")

From a conceptual point of view, there is no difference: you buffer the data, process it somehow, and send it on. But each type of buffer imposes its own constraints on the processing.
A pause for jargon

Data processing inside detector electronics (e.g. FPGAs) has a "fixed latency" => every bunch crossing has to be processed in the same amount of time to stay in sync.

Data processing in server farms does not have a fixed latency because data from all subdetectors has been aggregated ("event building"): the maximum average processing time is fixed, but busier events can take longer and emptier events less long.

Server farm does NOT mean CPUs. Could be a farm of CPUs, GPUs, FPGAs, CPU-GPU-FPGA hybrids…
When do you need fixed latency?

Whenever you cannot afford to read out all the information from all the subdetectors and build the “event” before processing it.
LHCb fixed latency calorimeter trigger

The problem naturally parallelizes: split CALO into regions, look for large clusters in each one. Keep event if one or more clusters passes a threshold.
LHCb fixed latency calorimeter trigger

The problem naturally parallelizes: split CALO into regions, look for large clusters in each one. Keep event if one or more clusters passes a threshold.
A cuter example: in HL-LHC CMS aim to reconstruct all charged particles with $p_T > 2$ GeV/c at 40 MHz. But tracks are not localized, can overlap…
However if you build your tracker with the right module spacing, pairs of nearby hits can allow you to locally select high-$p_T$ seeds! So designing fixed latency triggers is often closely linked to designing the detectors they use.
What about once the events are built?

Having read out the detector and assembled information from its different parts (subdetectors) into events, you still typically have too much data...
Where can you process your data then?

- In server farm near the detector ("software trigger")
- In server farm far from the detector ("GRID")

At this point the choice is where to process it further. Because of network cable costs, it is typically most cost-effective to build a custom server farm near the detector, but in the future minimizing power consumption may be more important and mandate large aggregated farms further away.
One thing which a server farm allows you to do is use the time between LHC fills to process data, so you can run a more complex data processing.
Of course you want to optimize that

The LHC actually has a broadly predictable behaviour, so use one year’s fill lengths to optimize the processing time in and out of fill for your farm.
Of course you want to optimize that

Notice that in this case individual events can hang around for weeks before finally being processed by the system. A very stretched kind of real-time.
And you can also use the GRID

You could in principle offload some of the work from your “near to the detector” server farm to some other server farm, on the GRID or even to a commercial cloud. You could also park the data (which ATLAS/CMS do in some cases) for some time and process it later when you have spare cycles. The only real question is having enough output bandwidth to send the rates required, and is it cost effective or not?
Recap: how real is time?

Fixed latency when data is too big to read out
Recap: how real is time?

- Fixed latency when data is too big to read out
- Fixed latency data processing typically in regions of interest closely linked to specific layout of detector
Recap: how real is time?

- Fixed latency when data is too big to read out.
- Fixed latency data processing typically in regions of interest closely linked to specific layout of detector.
- Once data can be read out, use farms of processors without fixed latency, including out of collision time.
Recap: how real is time?

- Fixed latency when data is too big to read out
- Fixed latency data processing typically in regions of interest closely linked to specific layout of detector
- Once data can be read out, use farms of processors without fixed latency, including out of collision time.
- But what kind of processing do we want to do?
Selection vs. compression in real-time data processing
Collisions at the LHC: summary

<table>
<thead>
<tr>
<th>Particle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proton</td>
<td>2804 bunch/beam</td>
</tr>
<tr>
<td>Protons/bunch</td>
<td>$10^{11}$</td>
</tr>
<tr>
<td>Beam energy</td>
<td>7 TeV ($7 	imes 10^{12}$ eV)</td>
</tr>
<tr>
<td>Luminosity</td>
<td>$10^{34}$ cm$^{-2}$ s$^{-1}$</td>
</tr>
<tr>
<td>Bunch crossing rate</td>
<td>30 MHz</td>
</tr>
<tr>
<td>Between 1-200</td>
<td>proton-proton collisions per crossing (depends on experiment).</td>
</tr>
<tr>
<td>New physics rate</td>
<td>$\approx 0.00001$ Hz</td>
</tr>
</tbody>
</table>

Event selection: 1 in $10,000,000,000,000,000$
Triggers
yesterday
But what if every collision is interesting?

Remember LHC collision rate is ~30 MHz

~10% of LHC collisions produce a charm-hadron pair, so events with multiple collisions are nearly all interesting if you want to study charm.
Triggers yesterday

Real-time data analysis today
Is every part of the collision interesting?

An event recorded by the LHCb detector. Even in the case that every event is interesting, do you need all of this event for your analysis?
Imagine your signal candidate are the two magenta tracks. You could save a lot of disk space & write more events if you only saved the magenta part!
Event selection vs. compression

**Event selection**

Is this event considered interesting by at least one algorithm ("trigger line")?

↓

**Write entire output of detector to permanent storage**
Event selection vs. compression

**Event selection**

Is this event considered interesting by at least one algorithm ("trigger line")?

Write entire output of detector to permanent storage

**Event compression**

Is this event considered interesting by at least one algorithm ("trigger line")?

Write signal candidate identified by trigger line to permanent storage
Why should this be a binary choice?

Write entire output of detector to permanent storage

Write signal candidate from real-time selection to permanent storage

In fact, what I presented as “selection” is just a special case of compression.
It isn’t a binary choice

A special case where “other interesting parts” is the entire event. Modern data processing (real-time or not) is a mixture of selection & compression.
Almost every analysis has two basic components:

1. Combine & select building blocks to observe a signal above background
2. Understand the efficiency of step (1) to measure signal properties
How do we understand the efficiencies?

Almost every analysis has two basic components:

1. Combine & select building blocks to observe a signal above background
2. Understand the efficiency of step (1) to measure signal properties
How does selection relate to efficiency?

Whenever you make a decision to keep/reject an object in your analysis, you must measure the efficiency of this decision for the object in question.
If keep or reject the whole event for future study based on the properties of the “signal” magenta tracks, you must only know the efficiency for them.
Or efficiency for selecting other objects

But if you keep only the magenta tracks plus some additional “interesting” objects, you must know the efficiency for every “interesting” object!
So there is a kind of binary aspect to this

- Write entire output of detector to permanent storage
- Write signal candidate from real-time selection & other interesting parts of event to permanent storage
- Write signal candidate from real-time selection to permanent storage
Either end: understand only signal

Write entire output of detector to permanent storage

Write signal candidate from real-time selection & other interesting parts of event to permanent storage

Write signal candidate from real-time selection to permanent storage

You must understand the efficiency of the criteria for your signal
But in the middle, understand more

- Write entire output of detector to permanent storage
- Write signal candidate from real-time selection & other interesting parts of event to permanent storage
- Write signal candidate from real-time selection to permanent storage

You must understand the efficiency of both the criteria for your signal AND the efficiency of the criteria for any other objects which you consider interesting!

You must understand the efficiency of the criteria for your signal
Recap: types of data processing

The goal of the data processing is to reduce the data volume to a level which is manageable for analysis.
Recap: types of data processing

The goal of the data processing is to reduce the data volume to a level which is manageable for analysis.

Traditionally this meant using criteria to select & record a small subset of “interesting” events for later analysis.
Recap: types of data processing

The goal of the data processing is to reduce the data volume to a level which is manageable for analysis.

Traditionally this meant using criteria to select & record a small subset of “interesting” events for later analysis.

We can also compress events, selecting & recording only those parts deemed interesting for analysis. This makes understanding selection efficiencies more critical.
Recap: types of data processing

The goal of the data processing is to reduce the data volume to a level which is manageable for analysis.

Traditionally this meant using criteria to select & record a small subset of “interesting” events for later analysis.

We can also compress events, selecting & recording only those parts deemed interesting for analysis. This makes understanding selection efficiencies more critical.

How do we optimize our finite processing budget?
Balancing the constraints on real-time reconstruction
What kinds of constraints?

Take LHCb reconstruction as an example, remember that ATLAS & CMS have much more complicated events. Need to reconstruct 1 million events/s.
Limited size of processing server farm

Take LHCb reconstruction as an example, remember that ATLAS&CMS have much more complicated events. Need to reconstruct 1 million events/s.
So the reconstruction must make choices.

Take LHCb reconstruction as an example, remember that ATLAS&CMS have much more complicated events. Impossible to fully reconstruct in real-time.

~50000 CPUs in parallel ~50 msec/event available

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track finding</td>
<td>~200 ms</td>
</tr>
<tr>
<td>Full track fit</td>
<td>~100 ms</td>
</tr>
<tr>
<td>RICH reconstruction</td>
<td>~180 ms</td>
</tr>
<tr>
<td>Calo reconstruction</td>
<td>~50 ms</td>
</tr>
<tr>
<td>Muon ID</td>
<td>~2 ms</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>~650 ms</td>
</tr>
</tbody>
</table>
So how do you optimize these choices?

Could be a lecture course in itself! Different analyses use different reconstructed objects, balance depends strongly on the physics programme of the experiment.

Will illustrate three concepts here

1. Selecting events can create time
2. Reconstructing more “expensive” event features can reduce the time cost of later reconstruction
3. Saving time by applying selection in reconstruction
How does a selection create time?

1 MHz of events

Fast/simple reconstruction
How does a selection create time?

1 MHz of events

Fast/simple reconstruction

Cost = X ms * 10⁹ events/ms
How does a selection create time?

1 MHz of events

Fast/simple reconstruction

Event selection stage

Cost = X ms * 10⁹ events/ms

90% Rejected
How does a selection create time?

- Fast/simple reconstruction
  - Cost = X ms * 10^9 events/ms

- Event selection stage
  - Selected
  - 90% Rejected
  - Cost = 10X ms * 10^8 events/ms

- More complex/comprehensive reconstruction stage

Most real-time reconstructions are in fact reconstruction-selection cascades.
Can expensive reconstruction save time?

Let’s say you want to reconstruct a $\Lambda_c \rightarrow pKK$ decay in your detector.
Can expensive reconstruction save time?

The signal are three charged particles displaced from the pp collision which vertex at the right mass. A typical LHC collision produces ~30 charged particles so ~25000 vertex combinations to fit (which takes time).
Can expensive reconstruction save time?

But most of the particles produced are pions, which we are not interested in! If we can select protons and kaons before vertexing, we can save time.
Can expensive reconstruction save time?

Particle identification is not cheap, but neither is vertexing many combinations. The balance will depend on the analysis\&detector of course.

Figure 7: Lifetime acceptance function for an event of a two-body hadronic decay. The shaded, light blue regions show the bands for accepting a track \( IP \). After \( IP_2 \) is too low in (a) it reaches the accepted range in (b). The actual measured lifetime lies in the accepted region (c), which continues to larger lifetimes (d).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track finding</td>
<td>~ 200 ms</td>
</tr>
<tr>
<td>Full track fit</td>
<td>~ 100 ms</td>
</tr>
<tr>
<td>RICH reconstruction</td>
<td>~ 180 ms</td>
</tr>
<tr>
<td>Calo reconstruction</td>
<td>~ 50 ms</td>
</tr>
<tr>
<td>Muon ID</td>
<td>~ 2 ms</td>
</tr>
<tr>
<td>VERTEXING</td>
<td>~ 120 ms</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>~ 650 ms</td>
</tr>
</tbody>
</table>
Let’s consider the LHCb detector and imagine that your signal always produces a charged particle with at least 1 GeV of $p_T$. 
Applying selections in the reconstruction

First you reconstruct particles in the vertex detector where there is no magnetic field, so this part is fast. But it does not give you the momentum.
Now you need to extend this particle through the tracking system, but this can be very slow because you don’t know how much the magnet bent it.
Applying selections in the reconstruction

But you know the smallest $p_T$ which it must have to pass your selection. This allows you to narrow the search window and gain a lot of time!
Recap: optimizing the reconstruction

Detector reconstruction is limited by the processing power and usually cannot run at full the collision rate.
Recap: optimizing the reconstruction

- Detector reconstruction is limited by the processing power and usually cannot run at full the collision rate.
- Using a simple reconstruction to select events can “buy” time to run more complex reconstructions.
Recap: optimizing the reconstruction

Detector reconstruction is limited by the processing power and usually cannot run at full the collision rate.

Using a simple reconstruction to select events can “buy” time to run more complex reconstructions.

Performing more complex reconstruction can save time by allowing a more sophisticated selection upfront.
Recap: optimizing the reconstruction

Detector reconstruction is limited by the processing power and usually cannot run at full the collision rate.

Using a simple reconstruction to select events can “buy” time to run more complex reconstructions.

Performing more complex reconstruction can save time by allowing a more sophisticated selection upfront.

Look out for places where you can speed up a reconstruction by applying selection criteria inside it.
Recap: optimizing the reconstruction

- Detector reconstruction is limited by the processing power and usually cannot run at full the collision rate.

- Using a simple reconstruction to select events can “buy” time to run more complex reconstructions.

- Performing more complex reconstruction can save time by allowing a more sophisticated selection upfront.

- Look out for places where you can speed up a reconstruction by applying selection criteria inside it.

How is this information used to keep/reject events?
Inclusive vs. exclusive selections in real-time analysis
What do I mean by inclusive/exclusive?

signal $S = \{object_1, object_2, object_3, \ldots, object_n\}$

Any signal can be described as collection of reconstructed objects.
What do I mean by inclusive/exclusive?

signal $S = \{\text{object}_1, \text{object}_2, \text{object}_3, \ldots, \text{object}_n\}$

inclusive — $(\exists s \subset S : \text{condition})$
What do I mean by inclusive/exclusive?

Any signal can be described as a collection of reconstructed objects. An inclusive selection identifies the signal based on the properties of a subset of these objects, allowing that some may not have been reconstructed. An exclusive selection requires all the objects in order to identify the signal.

signal $S = \{\text{object}_1, \text{object}_2, \text{object}_3, \ldots, \text{object}_n\}$

inclusive — ($\exists s \subset S : \text{condition}$)

exclusive — ($\exists S : \text{condition}$)
The benefit of an inclusive selection

Inclusive selections help if you don’t fully know what you are looking for
The benefit of an inclusive selection

Inclusive selections help if you don’t fully know what you are looking for
The benefit of an inclusive selection

Inclusive selections help if you don’t fully know what you are looking for
The benefit of an inclusive selection

Inclusive selections help if you don’t fully know what you are looking for
It helps you look for a class of signals

Many new proposed particles leave missing energy in the detector for example.
It helps you look for a class of signals

Many new proposed particles leave missing energy in the detector for example. Many others involve isolated high-energy leptons.
Another benefit of being inclusive

The signal might be too complicated to fully reconstruct in real-time. Remember the trick with applying a pT cut in the reconstruction?
is you don’t have to find all its children

The smaller the $p_T$ you want to search for, the less time this trick gains you. That is a general rule, lower $p_T$ objects take longer to reconstruct.
You select the signal on its hardest child

If your signal decays into multiple objects, on average one of those objects will have quite low $p_T$, and the bigger the number of objects the more this is true. Being able to select the signal using only its hardest product helps
So why not use an inclusive selection?

If you can use it, an inclusive selection is always a great idea. But if you have too much signal, it becomes impossible to select it efficiently.
Example from LHCb upgrade simulation

Efficiency of Run-I inclusive bbar selection retuned for the LHCb upgrade. The fall-off is not because of background, but because of real b-hadrons which by definition cannot be inclusively separated from a specific signal.
In two ways. Firstly you can discriminate against other b-hadron decays better, so the rate goes down. Secondly you can now exploit event compression to reduce the event size and write more events.
Remember our earlier discussion about reducing the time of combining reconstructed objects into signal candidates? This is usually more severe for exclusive selections.

EXCLUSIVE: MUST AFFORD COMBINATORICS OR RECONSTRUCTION TO REDUCE THEM
Combinatorics & inclusive/exclusive

Remember our earlier discussion about reducing the time of combining reconstructed objects into signal candidates? This is usually more severe for exclusive selections than inclusive ones where only part of signal is built.
Can you use a mixture?

Absolutely and in fact you often will do exactly that: an inclusive selection for the first cascade stages where rate can be higher, exclusive later. This is in fact logical because the more complex reconstruction gives access to the additional information needed for the exclusive selection to be efficient.
You will often hear your colleagues say that unless the real-time selection is inclusive, you will not be able to develop new analysis ideas once the data is already taken.

This is absolutely correct.

However the other side of this is that for precision measurements where your selection rate is dominated by your signal, you will not be able to achieve the full physics potential (sensitivity) without exclusive real-time selections and without event compression.
Inclusive selections are great if signal is rare, and/or partially unknown, and/or expensive to fully reconstruct.
Recap: inclusive & exclusive selections

**Inclusive selections** are great if signal is rare, and/or partially unknown, and/or expensive to fully reconstruct.

**Exclusive selections** are great if signal is too abundant to select inclusively, and/or is cheap to fully reconstruct.
Recap: inclusive & exclusive selections

Inclusive selections are great if signal is rare, and/or partially unknown, and/or expensive to fully reconstruct.

Exclusive selections are great if signal is too abundant to select inclusively, and/or is cheap to fully reconstruct.

Often a cascade is used with an inclusive preselection using a cheap reconstruction which enables a more expensive reconstruction and (semi)-exclusive selection.
Recap: inclusive & exclusive selections

Inclusive selections are great if signal is rare, and/or partially unknown, and/or expensive to fully reconstruct.

Exclusive selections are great if signal is too abundant to select inclusively, and/or is cheap to fully reconstruct.

Often a cascade is used with an inclusive preselection using a cheap reconstruction which enables a more expensive reconstruction and (semi)-exclusive selection.

So how to calibrate & understand this processing?
Calibrating the reconstruction and selection and understanding their performance
Almost every analysis has two basic components:

1. Combine & select building blocks to observe a signal above background.
2. Understand the efficiency of step (1) to measure signal properties.
Can’t you just use detector simulation?

If detector simulation matches data perfectly, just process the simulation identically to data and obtain the efficiencies that way. *It often doesn’t.*
Examples of hard problems for simulation

- Occupancies near the beampipe or around magnets
- Shower shapes and ageing in the calorimeters
- If the real-time reconstruction and selection evolve over time, it can be hard to simulate the correct mixture
- Momentum and pseudorapidity spectra, especially of light particles in the event

In general therefore, we will need to use data-driven ways to figure out what our efficiency was and how to correct the simulation to match data.
Better performance is easier to calibrate

All other things being equal, higher efficiencies are easier to calibrate than lower ones, and better detector resolutions easier than worse ones.
All other things being equal, higher efficiencies are easier to calibrate than lower ones, and better detector resolutions easier than worse ones. This is because simulation is almost always optimistic, so the better the intrinsic performance the smaller the systematic this can generate in your analysis.
This is the ratio of electron vs. muon rates at the first (fixed latency) level of the LHCb real-time selection in 2012. It changed significantly during the year mainly because of calorimeter ageing, so hard to simulate an average.
In 2015 LHCb introduced a new procedure for following the ageing of the calorimeter, which led to much more stable ratios of electron to muon rates. The jumps you see are deliberate selection changes, not ageing.
Aligning a detector can be slow, but you can speed it up by parallelizing the alignment process across a compute farm. Then you have to select the right events to feed the alignment algorithms, depending on the detector.
Consider a two-variable BDT: this is like a binned selection where the BDT algorithm picks the optimal bin sizes and boundaries. Make selections simpler to calibrate.
If you pick the binning yourself based on detector resolution and observed variations in the detector performance over time, you will lose some 5-10% of overall discriminating power but you get a selection which is simple(r) to calibrate and much faster to implement (becomes 1D lookup table).

by matching selection & reconstruction

Gligorov & Williams
http://arxiv.org/abs/1210.6861
You could also use classifiers which can be pre-calibrated to have a uniform efficiency or background rejection with respect to any variable/feature of interest for a marginal (few percent) loss in absolute performance. This is especially powerful and useful when you are selecting the signal with multivariate classifiers, which can distort kinematic/geometric distributions if not handled with care.
So which efficiency do you need to know?

That depends on what you are selecting and indeed compressing!
The common object efficiency?

You may need to know efficiency to reconstruct the basic common objects
The composite object efficiency?

You may need to know efficiency to make more complex common objects.
The signal selection efficiency?

- Raw data from detector
  - Common processing
  - Common selection of signal candidates and/or event compression
    - Signal candidates
    - Additional event information/objects

- Charged particle trajectories
  - π/K/p/e/μ

- Neutral clusters
  - γ/e/π⁰/K⁰/…

- Jets
  - τ/D⁰/MET/…

- Composite objects

And/or you may need to know the efficiency to perform the final selection.
The most basic technique which you can use for determining efficiencies is called tag&probe. You select the “probe” object for which you want to measure an efficiency using a separate “tag” object, and then count how frequently (efficiently) you also select the probe.
Let’s take as an example the efficiency of charged particle reconstruction (“tracking”). One of the most common tag-and-probe pairs is $J/\psi \rightarrow \mu \mu$, because muons are rare so the tag can be selected cleanly.
You of course need to partially select the probe, because otherwise how can you tell the tag muon came from a $J/\psi$? But you don’t need much resolution to see a mass peak so typically you can simply reconstruct the muon track in the muon detector, and rely on the standalone momentum measurement to get a mass peak (for the LHCb peak shown it is a bit more complicated but the idea is the same).
If you need to know the full efficiency of a specific real-time selection for your signal, the most common technique is also a kind of tag&probe. Let’s take as an example LHCb’s fixed latency selection which requires either an energetic calorimeter cluster, or an energetic muon or dimuon.
If you need to know the full efficiency of a specific real-time selection for your signal, the most common technique is also a kind of tag&probe. Let’s take as an example LHCb’s fixed latency selection which requires either an energetic calorimeter cluster, or an energetic muon or dimuon. Coming back to our earlier dimuon event, what can be the tag?
If you need to know the full efficiency of a specific real-time selection for your signal, the most common technique is also a kind of tag&probe. Let’s take as an example LHCb’s fixed latency selection which requires either an energetic calorimeter cluster, or an energetic muon or dimuon. Coming back to our earlier dimuon event, what can be the tag? We could split the muons into a tag and probe.
If you need to know the full efficiency of a specific real-time selection for your signal, the most common technique is also a kind of tag&probe. Let’s take as an example LHCb’s fixed latency selection which requires either an energetic calorimeter cluster, or an energetic muon or dimuon. Coming back to our earlier dimuon event, what can be the tag? Or for the dimuon it could be the calorimeter cluster.
And if efficiencies depend on kinematics?

Most efficiencies you need will vary as a function of your kinematics. It is therefore important that your calibration (tag&probe) and signal samples have matching kinematics.
You need to match the calibration & signal

Most efficiencies you need will vary as a function of your kinematics. It is therefore important that your calibration (tag & probe) and signal samples have matching kinematics. If they don’t, the calibration is not of much use.
This mismatch problem can also occur because the simulation kinematics don’t match the signal kinematics. In both cases you need to reweight the calibration or simulation samples to match your signal before using them. But if you are looking for a new beyond standard model signal, how do you know its kinematics in data?
You need to use a proxy

You don’t, but you can use a related well-known SM control channel which you are able to select cleanly, and derive data/simulation correction factors from this to port to your signal.
Some boobytraps when reweighting

If using an exclusive selection strategy you have to remember to write selections for all the control and calibration signals as well!

Be careful with kinematic regions which are not covered by your calibration samples. If your correction factors are close to 1, it may be safe to use a nearest-neighbour region as a proxy and assign a generous systematic uncertainty. If not consider removing them.

If the signal and control samples are very different from each other and their distributions vary rapidly, you need to bin more finely when reweighting in order to avoid biases.
Say you are calibrating data-simulation differences and get the above plot for efficiency on data/simulation. Would you feel comfortable extrapolating into the region not covered in the data sample?
Say you are calibrating data-simulation differences and get the above plot for efficiency on data/simulation. Would you feel comfortable extrapolating into the region not covered in the data sample? What about now?
Say you are calibrating data-simulation differences and get the above plot for efficiency on data/simulation. Would you feel comfortable extrapolating into the region not covered in the data sample? What about now?
Another pitfall with a binned plot is that each entry is the average efficiency in a given bin. But this assumes signal and control samples have roughly the same kinematics in each bin!
Danger of rapidly varying distributions

Another pitfall with a binned plot is that each entry is the average efficiency in a given bin. But this assumes signal and control samples have roughly the same kinematics in each bin! If not, bin more finely or reweight.
Recap: calibrating and understanding

Unless your simulation matches data perfectly, you will need to calibrate your reconstruction & selection.
<table>
<thead>
<tr>
<th>Recap: calibrating and understanding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unless your simulation matches data perfectly, you will need to calibrate your reconstruction &amp; selection</td>
</tr>
<tr>
<td>If the detector &amp; reconstruction are inherently more efficient, stable, and well aligned, this will be easier</td>
</tr>
</tbody>
</table>
Recap: calibrating and understanding

Unless your simulation matches data perfectly, you will need to calibrate your reconstruction & selection.

If the detector & reconstruction are inherently more efficient, stable, and well aligned, this will be easier.

Most calibrations rely on tag & probe techniques to measure efficiencies on data, and reweighting to make the simulation & control samples look like your signal.
Recap: calibrating and understanding

Unless your simulation matches data perfectly, you will need to calibrate your reconstruction & selection.

If the detector & reconstruction are inherently more efficient, stable, and well aligned, this will be easier.

Most calibrations rely on tag & probe techniques to measure efficiencies on data, and reweighting to make the simulation & control samples look like your signal.

Thanks for listening!