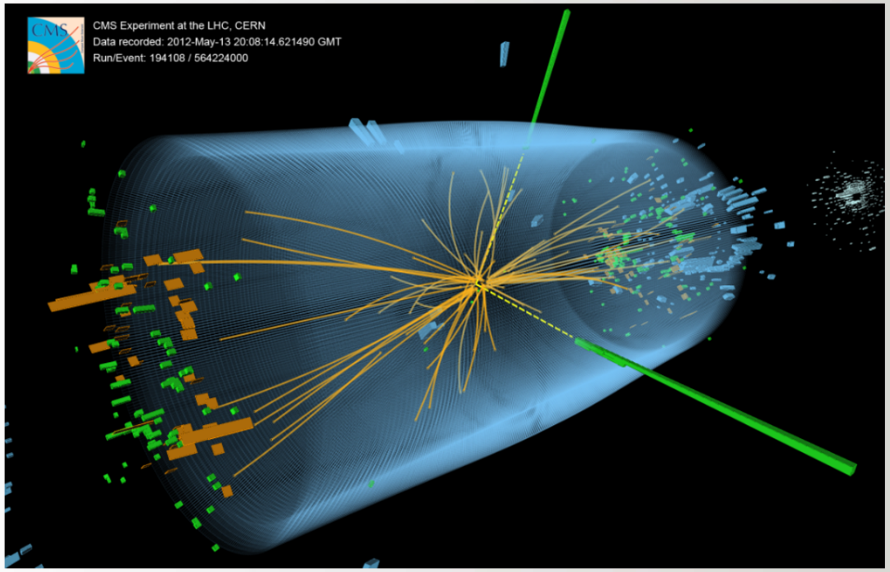
How to discover the Higgs Boson in an Oracle database

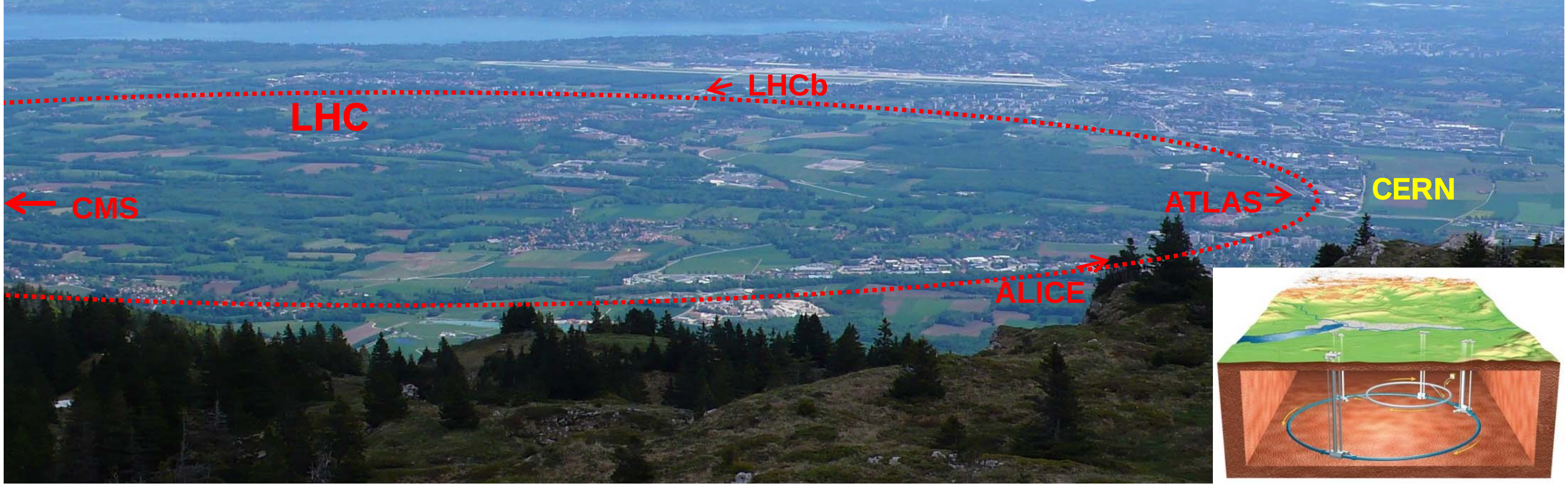
Maaike Limper



CERN openlab

Large Hadron Collider at CERN

- The Large Hadron Collider is used to collide hadrons (protons or lead ions) at high energy, it is currently the world's most powerful particle accelerator
- Four main experiments placed along the ring study the events produced by the Large Hadron Collider: ATLAS, CMS, LHCb and ALICE



Introduction

*"**CERN openlab** is a unique public-private partnership between CERN and leading ICT companies. Its mission is to accelerate the development of cutting-edge solutions to be used by the worldwide LHC community"* http://openlab.web.cern.ch

In January 2012 I joined Openlab as an Oracle sponsored CERN fellow

My project: *Investigate the possibility of doing LHC-scale data analysis within an Oracle database*

Introduction

Some of the items I will discuss with you today:

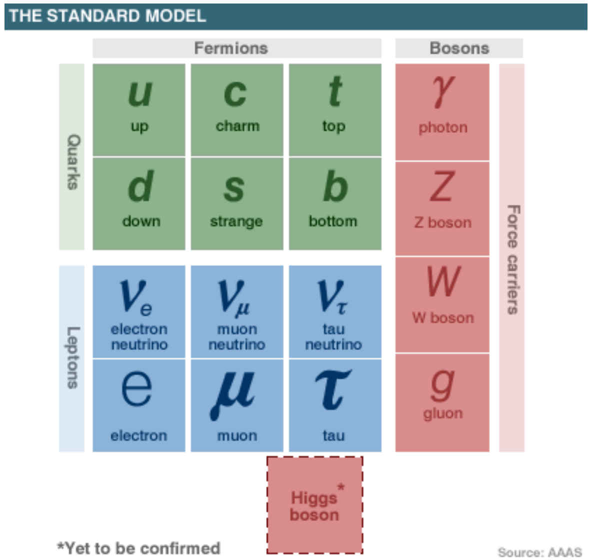
- **Data processing at CERN: how do we go from detector measurements to discovering new particles**
- **An example of a database structure containing analysis data**
- **An example of physics analysis code converted to SQL**
- **Using outside algorithms (C++/java) as part of the event selection**
- **Outlook: how to scale my studies to real LHC-scale data analysis**

***Disclaimer: any results shown today are for the purpose of illustrating my studies and are by no means to be interpreted as real physics results*!**

A quick course in Particle Physics

The Standard Model (SM) of particle physics, a very successful theory describing all the elementary particles and forces that are the building blocks of our world

- **Leptons: electrons and muons, relatively easy to detect, the weapon of choice for many physics analysers!** (tau's are tough)
- **Neutrino's can not be detected directly**



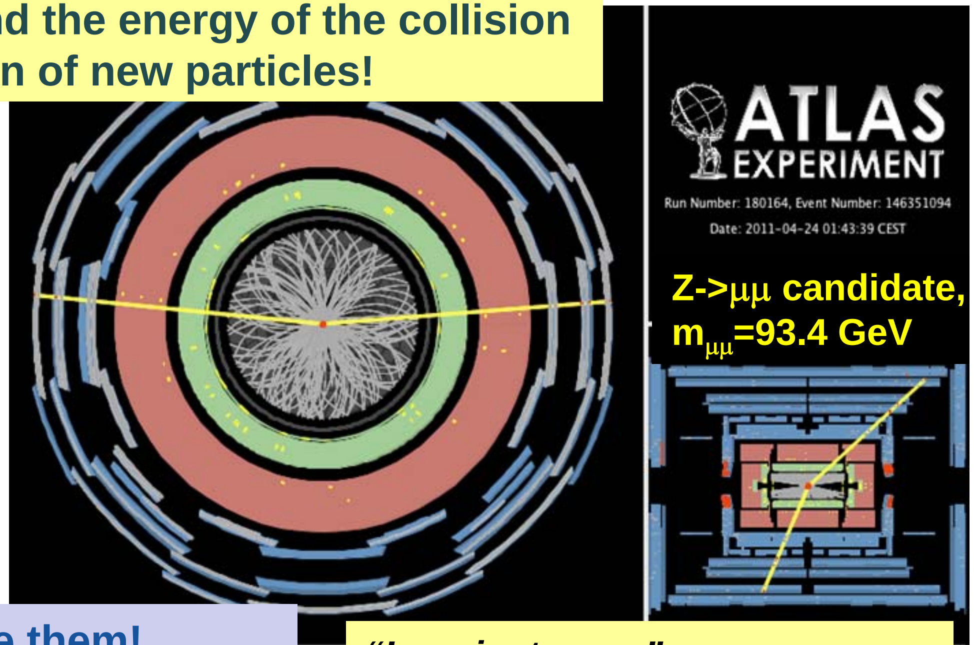
- **Quarks: can not exist alone, but are bound together in pair or triplets**
- **Particles composed of quarks are called hadrons, for example, proton=two up plus one down quark**

- **Force carriers: photons for electro-magnetism, Z/W bosons for 'weak' interaction, gluons to bind quarks together and the Higgs boson to give mass to particles**

A quick course in Particle Physics

When the Large Hadron Collider collides protons at high energy the particles interact and the energy of the collision is converted into the production of new particles!

- **The detectors built around the collision point measure the produced particles**
- **high energy quark production results in a 'jet' of particles seen in the detector**
- energy resulting from a collision at the LHC is spread symmetrically, an imbalance in the energy measured by the detectors often indicate the presence of neutrino's in the event



Many particles decay before we can measure them!

Instead we see these by their "invariant mass" calculated from the energy and momentum of the decay products

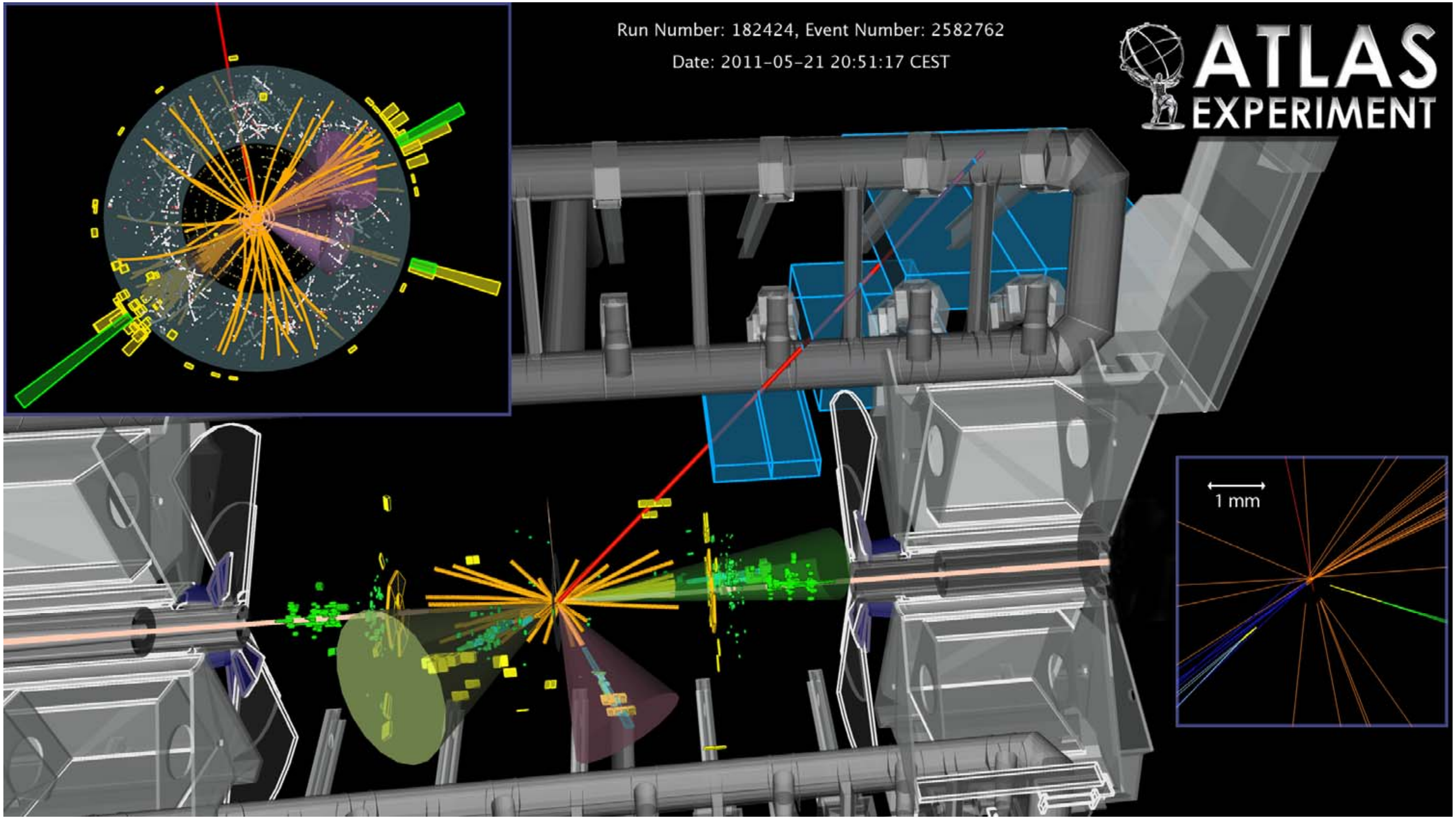
"Invariant mass"

$$Mc^2=(\sum E)^2 + \|\sum \vec{p}c\|^2$$

M=invariant mass, equal to mass of decay particle

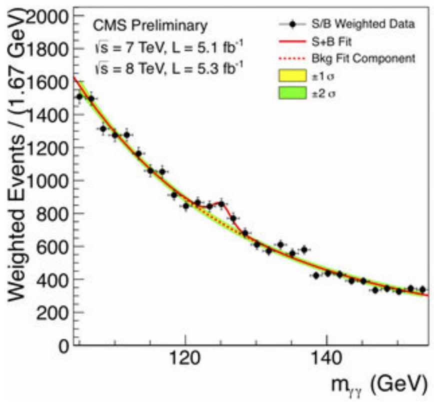
$\sum E$=sum of the energies of produced particles

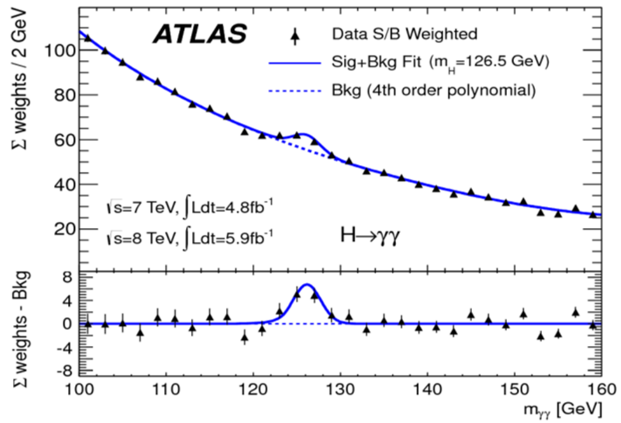
$\|\sum \vec{p}c\|$=vector sum of momenta of produced particles


Run Number: 182424, Event Number: 2582762
Date: 2011-05-21 20:51:17 CEST
ATLAS
EXPERIMENT
1 mm

Discovery of a "Higgs boson-like" particle

Plots of the invariant mass of photon-pairs produced at the LHC show a significant bump around 125 GeV





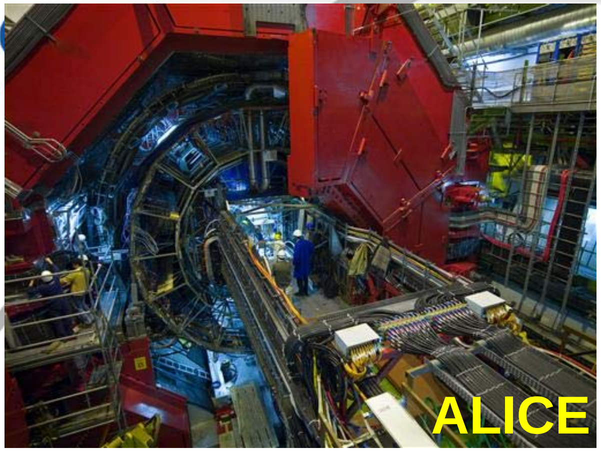
The discovery of a "Higgs boson-like" particle!

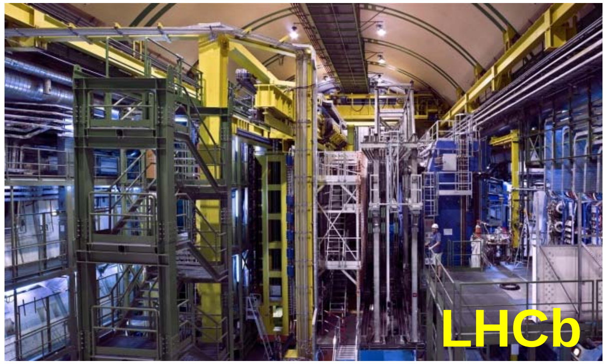
http://www.bbc.co.uk/news/world-18702455

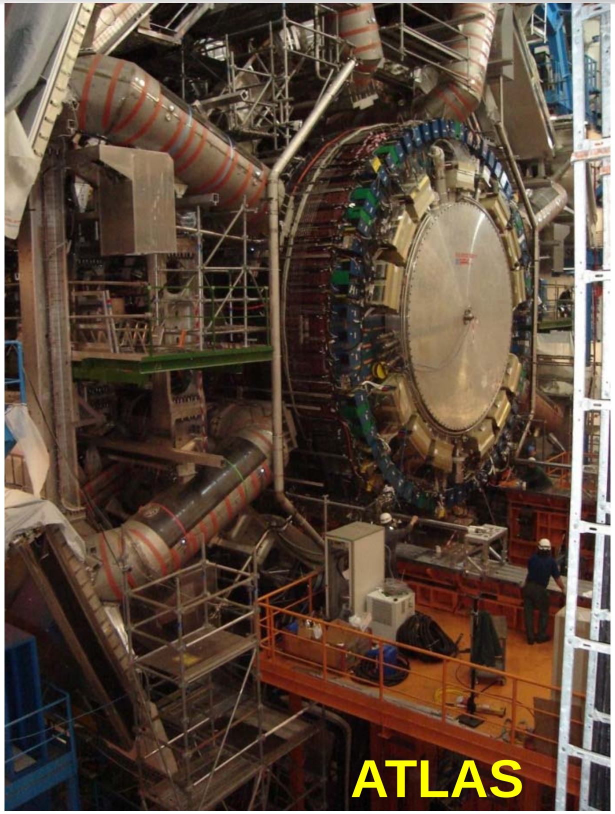
- The work of thousands of people!
- Operations of LHC and its experiments rely on databases for storing conditions data, log files etc.

… but the data-points in these plots did not came out of a database !

Where does the data come from?

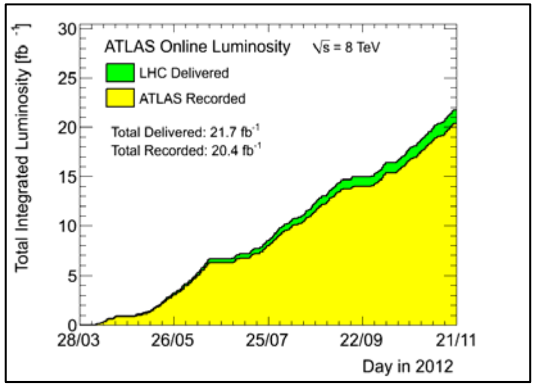


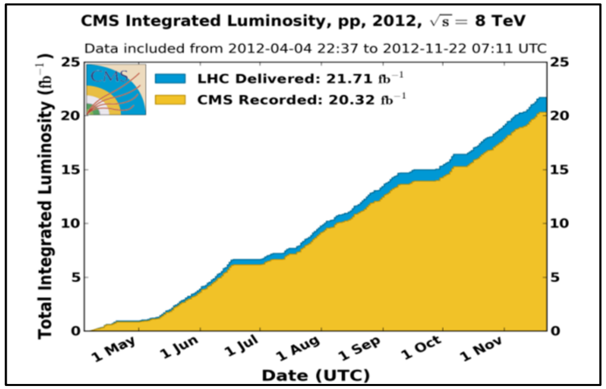




Where does the data come from?

Large amounts of "high luminosity" data recorded by the experiments:





The LHC produces 40 million proton-proton collision events per second

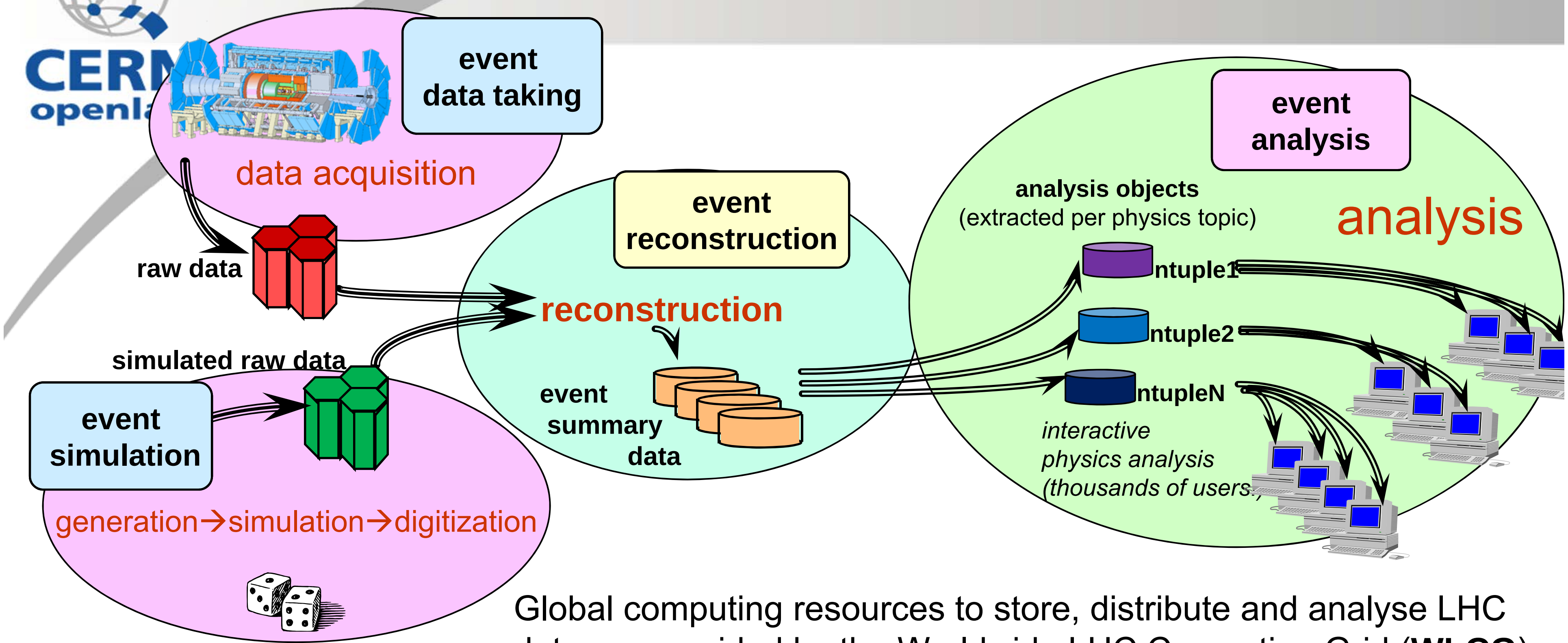
Not all events are recorded, trigger electronics built into the detectors help determine which events are interesting enough to keep

Some recent numbers from the ATLAS experiment in 2012:

~400 events recorded per second during an LHC run

~2 billion events recorded, ~2 PB of raw data, ~3 PB of Event Summary Data

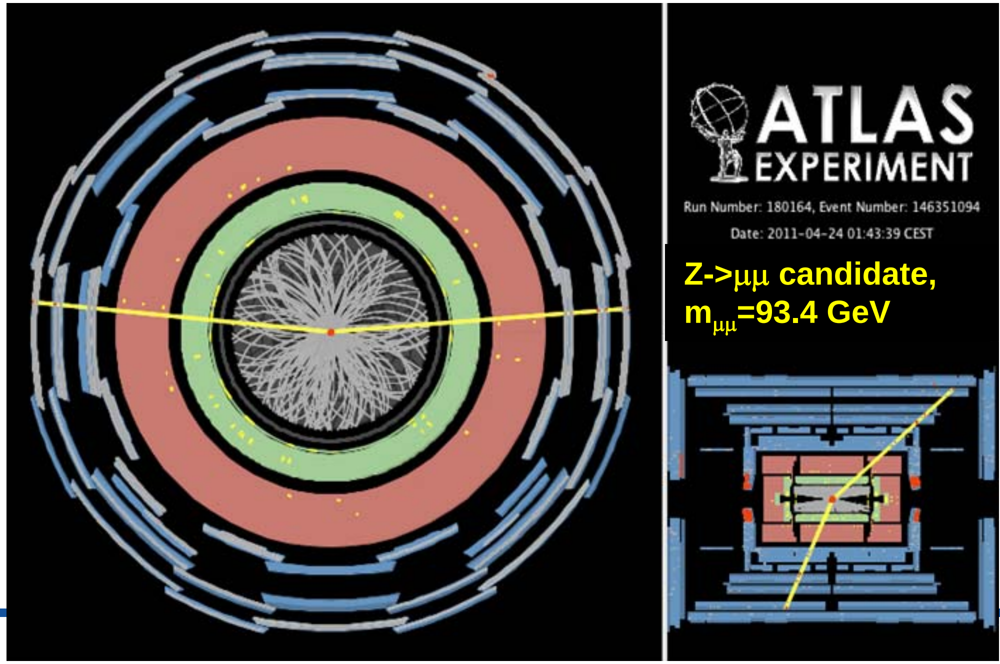
Where does the data come from?



Global computing resources to store, distribute and analyse LHC data are provided by the Worldwide LHC Computing Grid (**WLCG**) which has more than than 170 computing centres in 36 countries

Analysis versus reconstruction

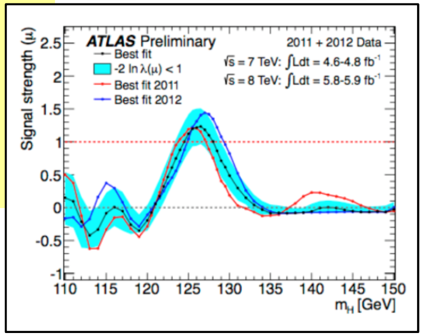
Event Reconstruction focuses on creating physics objects from the information measured in the detector
Event Analysis focuses on interpreting information from the reconstructed objects to determine what type of event took place



Data analysis in practice

LHC Physics analysis is done with ROOT

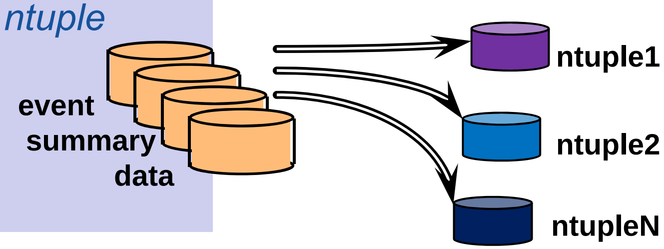
- Dedicated C++ framework developed by the High Energy Physics community, http://root.cern.ch
- Provides tools for plotting/fitting/statistic analysis etc.



ROOT-ntuples are centrally produced by physics groups from previously reconstructed event summary data

Each physics group determines specific content of ntuple

- *Physics objects to include*
- *Level of detail to be stored per physics object*
- *Event filter and/or pre-analysis steps*

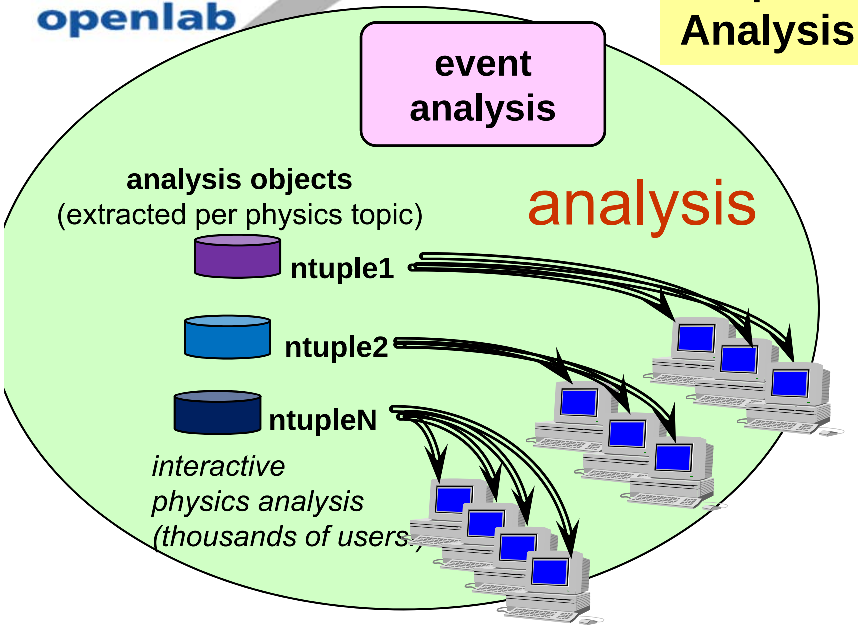


Data is stored as a "TTree" object, with a "TBranch" for each variable

Variables for each event in the form of scalar (number of muons), vectors (energy of each muon), vector-of-vectors (position of each detector hit for each muon)

Data analysis in practice

Ntuples are centrally produced per physics topic
Analysis is typically I/O intensive and runs on many files



Small datasets→ copy data and run analysis locally

Large datasets:→use the LHC Computing Grid

- **Grid computing tools split the analysis job in multiple jobs each running on a subset of the data**
- Each sub-job is sent to Grid site where input files are available
- Results produced summed at the end

Bored waiting days for all grid-jobs to finish→
Filter data and produce private mini-ntuples

Can we replace the ntuple analysis with a model where data is analysed from an Oracle database?

Data analysis in a database

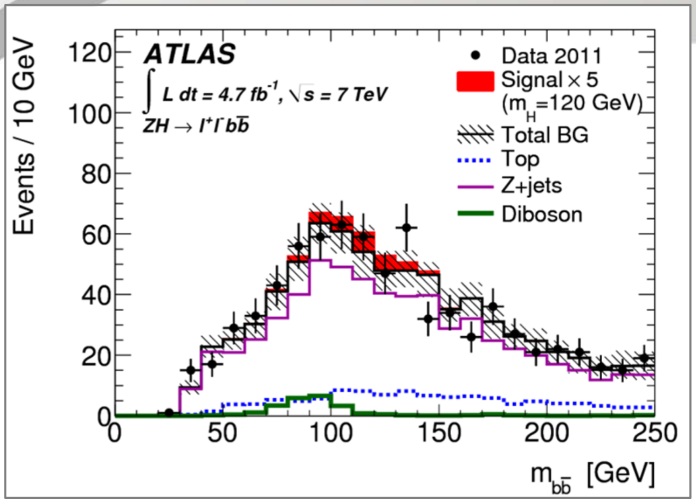
Benchmark Physics Analysis in an Oracle DB:

- **Simplified version of the HZ→bbll analysis** (search for standard model Higgs boson produced in association with a Z-boson)
  - **Select lepton-candidates to reconstruct Z-peak**
  - **Select b-jet-candidates to reconstruct Higgs-peak**

Oracle database filled with data from two samples of simulated data:

- **Signal sample: 30 k events (3 ntuples)**
- **Background sample (Z+2/3/4 jets): 1662 k events (168 ntuples)**
- Use ntuple defined by ATLAS Top Physics Group: ”NTUP_TOP”
  - 4212 physics attributes per event
  - Size of each ntuple is approx. 850 MB

HZ→bbll analysis



Plot from ATLAS published result, full paper at: [*http://arxiv.org/pdf/1207.0210.pdf*](http://arxiv.org/pdf/1207.0210.pdf)

For my studies I used simulation data produced for the <u>"HZ→bbll" analysis</u>

"The search for SM Higgs boson produced in association with a Z-boson":

- *A SM Higgs boson with a mass 125 GeV decays mainly to two b-quarks*
- *An event topology with only two b-jets has a large background from jet production*
- *If the Higgs boson is produced together with a Z-boson and the Z decays to leptons, the events are easier to detect and the background becomes significantly less*

Physics Database implementation

Currently implemented 1042 variables, divided over 5 different tables

Separate schema for each sample

Variable "EventNo_RunNo" uniquely defines each event

Tables "eventData" and "MET"(missing transverse energy):

- One row of data for each event
- primaryKey=(EventNo_RunNo)

Tables "muon", "electron" and "jet":

- One row of data for each muon/electron/jet object
- primaryKey=(muonId/jetID/electronID,EventNo_RunNo),
- "EventNo_RunNo" is indexed

Table statistics:

ZH->llbb

Table name	*columns*	*k rows*	*k blocks*	*size in MB*
MET	56	30	2.15	17
eventData	185	30	2.73	21
muon	297	38	12.4	97
electron	305	223	69.08	540
jet	210	481	107.36	839

Z->ll + 2/3/4 jets

Table name	*columns*	*k rows*	*k blocks*	*size in MB*
MET	56	1662	119.44	933
eventData	185	1662	151.13	1181
muon	297	1489	481	3758
electron	305	10971	3274.72	25584
jet	210	27931	5943.19	46431

Physics Analysis (1)

The goal of the analysis is to select signal events and removing as many background events as possible

The ratio of signal over background events will determine the significance of your discovery!

My version of the HZ→bbll analysis

- **MET selection:** Missing tranverse energy in events less then 30 < GeV
- **electron selection:** "IsElectron"-function to return TRUE, include requirement p_T>20 GeV and |η|<2.4 plus several requirement on hits and holes on tracks
- **muon selection:** "IsMuon"-function to return TRUE, include requirement p_T>20 GeV and |η|<2.4 plus several requirement on hits and holes on tracks
- **Require exactly 2 selected muons OR 2 selected electrons per event**
- **b-jet selection:** tranverse momentum greater than p_T>25 GeV, |η|<2.5 and "flavour_weight_Comb">1.55 (to select b-jets)
- Require opening-angle between jets ΔR>0.7 when p_TH< 200 MeV
- Require exactly 2 selected b-jets per event
- Require 1 of the 2 b-jets to have p_T>45 GeV
- **Plot "*invariant mass*" of the leptons (Z-peak) and of the b-jets (Higgs-peak)**

My analysis uses a total of 40 different variables from "MET", "jet", "muon" and "electron" tables

Database versus ntuples

Two versions of my analysis:

1. **Standard ntuple-analysis in ROOT (C++) using locally stored ntuples**
   - load only the branches needed for the analysis to make the analysis as fast as possible
   - loop over all events and applies the selection criteria event-by-event
2. **Analysis from the same data stored in the Oracle database** using functions for invariant mass and lepton selection implemented in PL/SQL
   - Executes a single SQL-query performing the data analysis via TOracleServer-class in ROOT
   - Rows returned by the query via TOracleServer are used to produce histograms

Check that both methods produce the same result and see which is faster!

Physics Analysis (1) SQL (part 1)

```
with sel_MET_events as (select /*+ MATERIALIZE FULL("MET_LocHadTopo") */
"EventNo_RunNo","EventNumber","RunNumber" from "MET_LocHadTopo" where
PHYSANALYSIS.pass_met_selection("etx","ety" ) = 1 ),
sel_electron as (select /*+ MATERIALIZE FULL("electron") */ "electron_i","EventNo_RunNo","E","px","py","pz" from "electron"
where PHYSANALYSIS.IS_ELECTRON("pt","eta","author","mediumWithTrack", 20000., 2.5) = 1 ),
sel_electron_count as (select "EventNo_RunNo",COUNT(*) as "el_sel_n" from sel_electron group by "EventNo_RunNo"),
sel_muon as (select /*+ MATERIALIZE FULL("muon") */  "muon_i","EventNo_RunNo","E","px","py","pz" from "muon"  where
PHYSANALYSIS.IS_MUON("muon_i", "pt", "eta", "phi", "E", "me_qoverp_exPV", "id_qoverp_exPV","me_theta_exPV",
"id_theta_exPV", "id_theta", "isCombinedMuon", "isLowPtReconstructedMuon","tight","expectBLayerHit", "nBLHits",
"nPixHits","nPixelDeadSensors","nPixHoles","nSCTHits","nSCTDeadSensors","nSCTHoles","nTRTHits","nTRTOutliers",0,20000.,
2.4) = 1 ),
sel_muon_count as (select "EventNo_RunNo",COUNT(*) as "mu_sel_n" from sel_muon group by "EventNo_RunNo" ),
sel_mu_el_events as (select /*+ MATERIALIZE */ "EventNo_RunNo","el_sel_n","mu_sel_n" from sel_MET_events LEFT
OUTER JOIN sel_electron_count USING ("EventNo_RunNo") LEFT OUTER JOIN sel_muon_count USING ("EventNo_RunNo")
where ("el_sel_n"=2 and "mu_sel_n" is NULL) or ("el_sel_n" is NULL and "mu_sel_n"=2) ),
```

List of selection criteria translates into a set of select statements defined as temporary tables

Without MATERIALIZE hint, query optimizer gets confused…

JOIN is used to combine information from different tables

FULL table scan is usually fastest, I'll come back to that later…

Physics Analysis (1) SQL (part 2)

sel_electron_events as (select /*+ MATERIALIZE */ "EventNo_RunNo",PHYSANALYSIS.INV_MASS_LEPTONS(el0."E",el1."E",el0."px",el1."px",el0."py",el1."py",el0."pz",el1."pz")/1000. as "DiElectronMass" from sel_mu_el_events INNER JOIN sel_electron el0 USING ("EventNo_RunNo") INNER JOIN sel_electron el1 USING ("EventNo_RunNo") where el0."electron_i"<el1."electron_i"),
sel_muon_events as (select /*+ MATERIALIZE */ “EventNo_RunNo",PHYSANALYSIS.INV_MASS_LEPTONS(muon0."E",muon1."E",muon0."px",muon1."px",muon0."py",muon1."py",muon0."pz",muon1."pz")/1000. as "DiMuonMass " from sel_mu_el_events INNER JOIN sel_muon muon0 USING ("EventNo_RunNo") INNER JOIN sel_muon muon1 USING ("EventNo_RunNo") where muon0."muon_i"<muon1."muon_i"),
sel_jet as (select /*+ MATERIALIZE FULL("jet") */ "jet_i","EventNo_RunNo","E","pt","phi","eta" from "jet" where "pt">25000. and abs("eta")<2. 5 and "fl_w_Comb">1.55),
sel_jet_count as (select "EventNo_RunNo" from sel_jet group by "EventNo_RunNo" HAVING MAX("pt")>45000. and COUNT(*) = 2),
sel_jet_events as (select /*+ MATERIALIZE */ "EventNo_RunNo",PHYSANALYSIS.INV_MASS_JETS(jet0."E",jet1."E",jet0."pt",jet1."pt",jet0."phi",jet1."phi",jet0."eta",jet1."eta")/1000. as "DiJetMass“ from sel_jet_count INNER JOIN sel_jet jet0 USING ("EventNo_RunNo") INNER JOIN sel_jet jet1 USING ("EventNo_RunNo") where jet0."jet_i"<jet1."jet_i" and PHYSANALYSIS.pass_bjet_pair_selection(jet0."pt"/1000.,jet1."pt"/1000.,jet0."phi",jet1."phi",jet0."eta",jet1."eta") = 1)

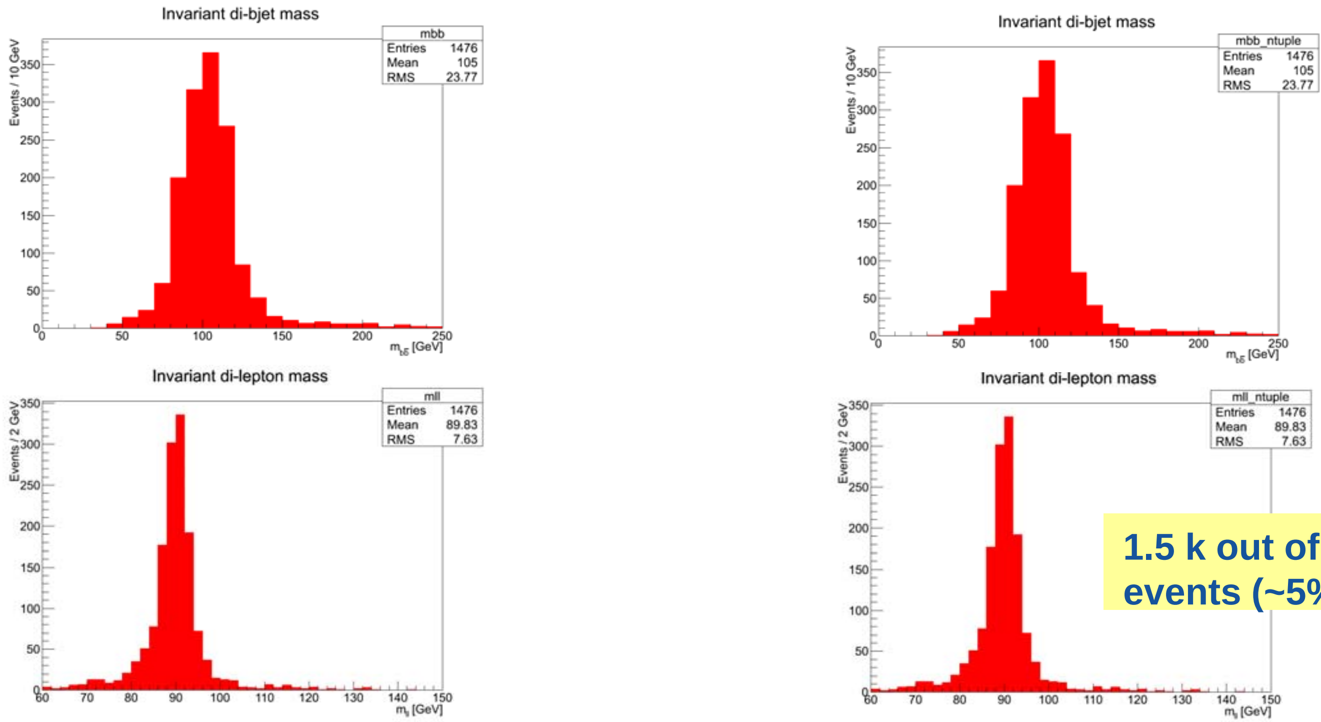
select "EventNo_RunNo","EventNumber","RunNumber",**"DiMuonMass","DiElectronMass","DiJetMass"** from sel_muon_events FULL OUTER JOIN sel_electron_events USING ("EventNo_RunNo") INNER JOIN sel_jet_events USING ("EventNo_RunNo") INNER JOIN sel_MET_events USING ("EventNo_RunNo")

The final select-statement returns the invariant mass of the leptons and jets

Plots Physics Analysis (1)

HZ→bbll sample

Database analysis | **Ntuple analysis**

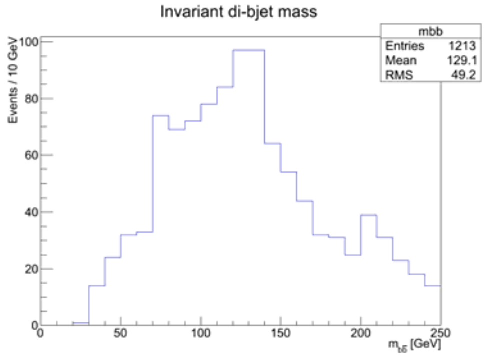


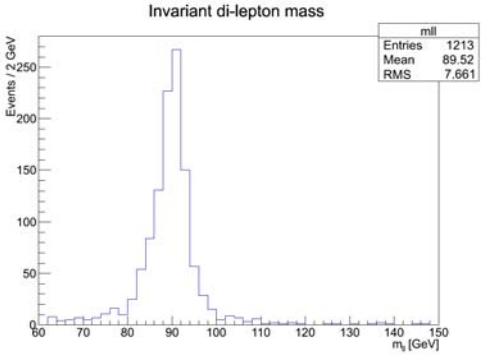
1.5 k out of 30 k events (~5%)

Plots Physics Analysis (1)

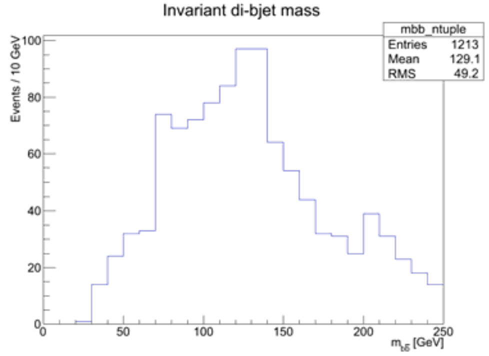
Z→ll+2/3/4 jets sample

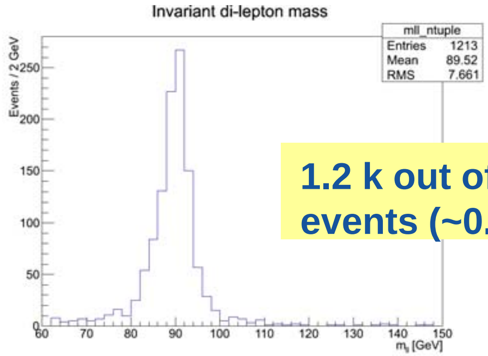
Database analysis





Ntuple analysis





1.2 k out of 1662 k events (~0.08%)

Timing Physics Analysis (1)

Timing results done after clearing caches for more consistent results

ntuple: sync && sysctl -w *vm.drop_caches=3*

DB: alter system flush buffer_cache; alter system flush shared_pool

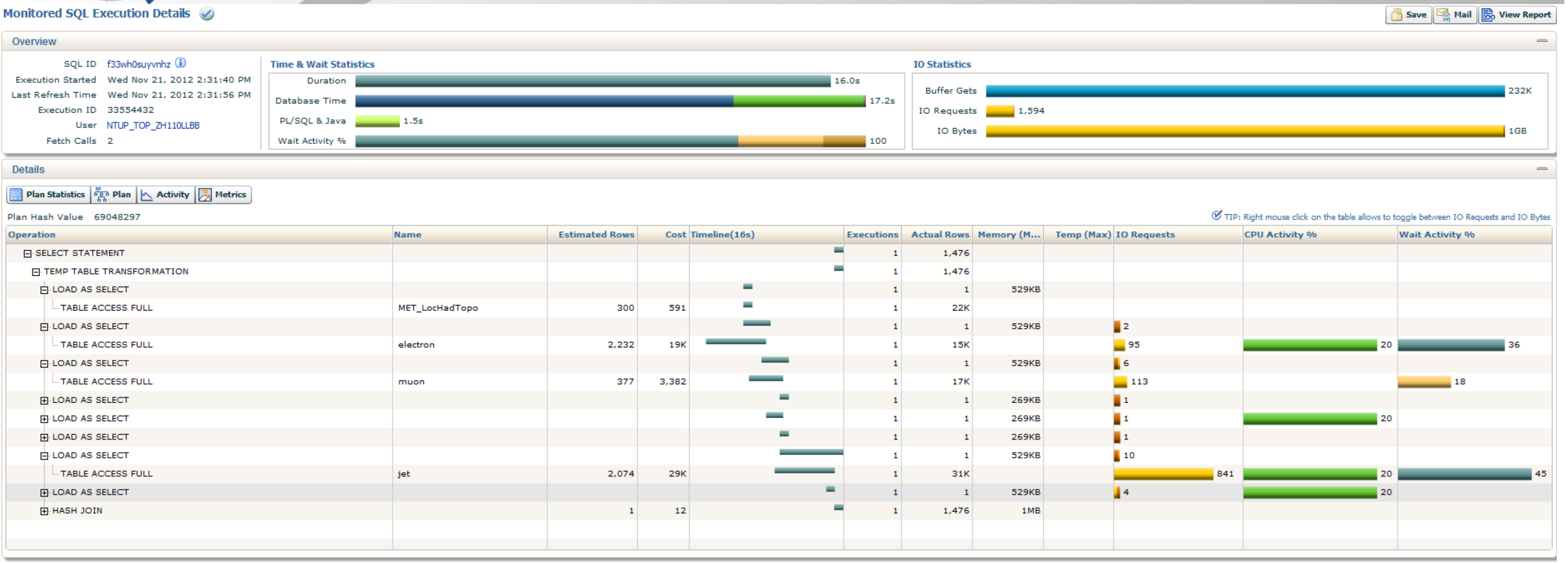
Database runs on the same machine as the root ntuple analysis

Ntuple-files are stored in the same disk-space as database-files

ZH→llbb sample:	
Ntuple analysis:	**12** seconds
Database analysis:	**18** seconds
Z→ll + jets sample:	
Ntuple analysis:	**508** seconds
Database analysis:	**333** seconds

For the large background sample the analysis from the database is faster than the ntuples analysis!

SQL monitoring
Physics Analysis (1) ZH→llbb



Monitored SQL Execution Details

Save | Mail | View Report

Overview

SQL ID	f33wh0suyvnhz
Execution Started	Wed Nov 21, 2012 2:31:40 PM
Last Refresh Time	Wed Nov 21, 2012 2:31:56 PM
Execution ID	33554432
User	NTUP_TOP_ZH110LLBB
Fetch Calls	2

Details

Plan Statistics | Plan | Activity | Metrics

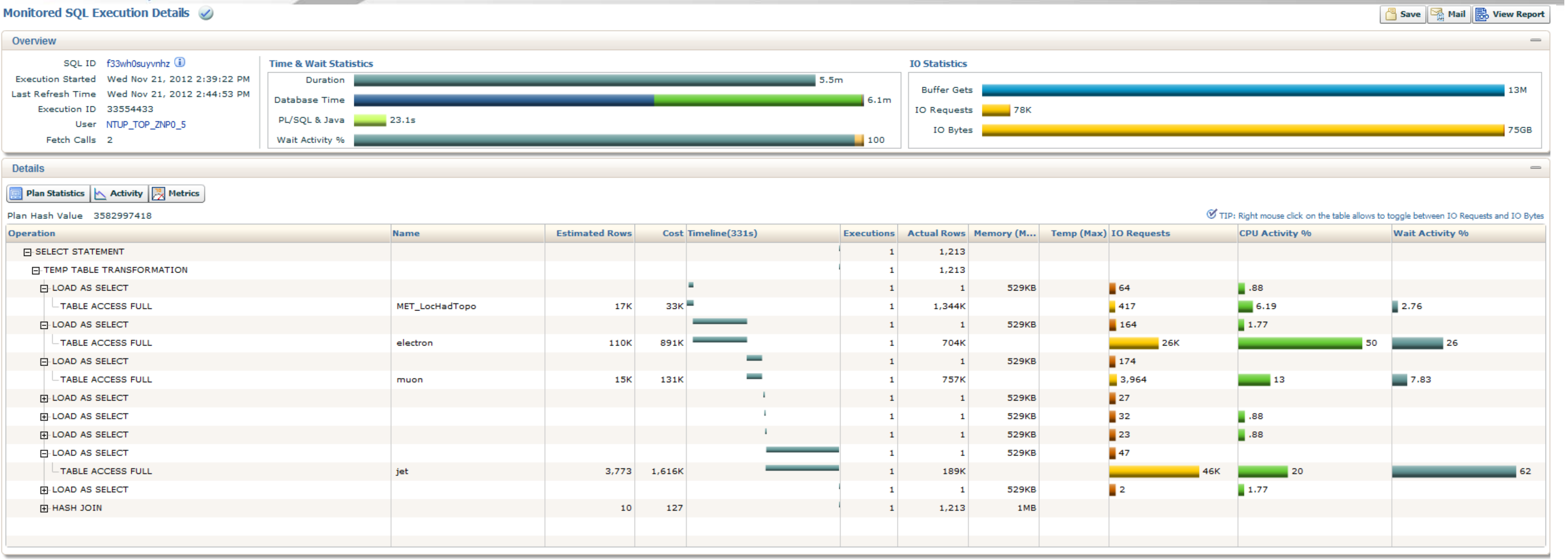
Plan Hash Value 69048297

TIP: Right mouse click on the table allows to toggle between IO Requests and IO Bytes

Operation	Name	Estimated Rows	Cost	Timeline(16s)	Executions	Actual Rows	Memory (M...	Temp (Max)	IO Requests	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	1,476					
TEMP TABLE TRANSFORMATION					1	1,476					
LOAD AS SELECT					1	1	529KB				
TABLE ACCESS FULL	MET_LocHadTopo	300	591		1	22K					
LOAD AS SELECT					1	1	529KB		2		
TABLE ACCESS FULL	electron	2,232	19K		1	15K			95	20	36
LOAD AS SELECT					1	1	529KB		6		
TABLE ACCESS FULL	muon	377	3,382		1	17K			113		18
LOAD AS SELECT					1	1	269KB		1		
LOAD AS SELECT					1	1	269KB		1	20	
LOAD AS SELECT					1	1	269KB		1		
LOAD AS SELECT					1	1	529KB		10		
TABLE ACCESS FULL	jet	2,074	29K		1	31K			841	20	45
LOAD AS SELECT					1	1	529KB		4	20	
HASH JOIN		1	12		1	1,476	1MB				

CPU: 4.5 s
IO-wait: 14.2 s
PL/SQL: 1.3 s

SQL monitoring
Physics Analysis (1) Z→ll+jets



Monitored SQL Execution Details

Save | Mail | View Report

Overview

SQL ID	f33wh0suyvnhz
Execution Started	Wed Nov 21, 2012 2:39:22 PM
Last Refresh Time	Wed Nov 21, 2012 2:44:53 PM
Execution ID	33554433
User	NTUP_TOP_ZNP0_5
Fetch Calls	2

Time & Wait Statistics

Duration	5.5m
Database Time	6.1m
PL/SQL & Java	23.1s
Wait Activity %	100

IO Statistics

Buffer Gets	13M
IO Requests	78K
IO Bytes	75GB

Details

Plan Statistics | Activity | Metrics

Plan Hash Value 3582997418

TIP: Right mouse click on the table allows to toggle between IO Requests and IO Bytes

Operation	Name	Estimated Rows	Cost	Timeline(331s)	Executions	Actual Rows	Memory (M...	Temp (Max)	IO Requests	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	1,213					
TEMP TABLE TRANSFORMATION					1	1,213					
LOAD AS SELECT					1	1	529KB		64	.88	
TABLE ACCESS FULL	MET_LocHadTopo	17K	33K		1	1,344K			417	6.19	2.76
LOAD AS SELECT					1	1	529KB		164	1.77	
TABLE ACCESS FULL	electron	110K	891K		1	704K			26K	50	26
LOAD AS SELECT					1	1	529KB		174		
TABLE ACCESS FULL	muon	15K	131K		1	757K			3,964	13	7.83
LOAD AS SELECT					1	1	529KB		27		
LOAD AS SELECT					1	1	529KB		32	.88	
LOAD AS SELECT					1	1	529KB		23	.88	
LOAD AS SELECT					1	1	529KB		47		
TABLE ACCESS FULL	jet	3,773	1,616K		1	189K			46K	20	62
LOAD AS SELECT					1	1	529KB		2	1.77	
HASH JOIN		10	127		1	1,213	1MB				

CPU: 154.1 s
IO-wait: 231.3 s
PL/SQL: 24.6 s

Physics Analysis (2)

Add an additional complication to the analysis:

- **Changed b-jet selection: recalculate the jet “flavour weight” for a better b-tagging performance**
- **“flavour_weight_Comb”>1.55 is now: mv1Eval(flavour_weight_IP3D, flavour_weight_SV1, flavour_weight_jetFitterCombNN, pt, eta)>0.60173**

“mv1Eval”: a neural-network based algorithm that combines the output of different b-tagging weights to calculate an optimized b-tagging weight→ **C++ code from the ATLAS experiment**

I’m too lazy/stupid to rewrite this algorithm in PL/SQL …

MV1 algorithm was written in C++, I can compile it and call it as an external:

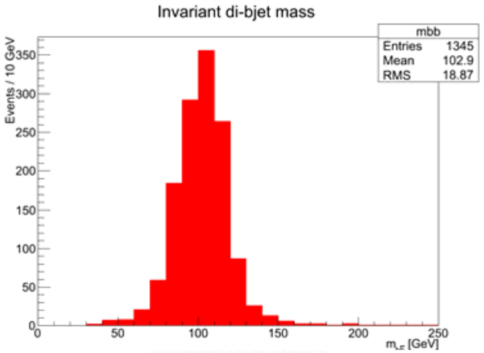
```
FUNCTION mv1Eval_fromExternal( w_IP3D double precision,  w_SV1 double precision, w_JetFitterCombNN double precision, jet_pt double precision, jet_eta double precision ) return double precision

 AS EXTERNAL library "MV1_lib" name "mv1Eval" language c parameters (w_IP3D double, w_SV1 double, w_jetFitterCombNN double, jet_pt double, jet_eta double);
```

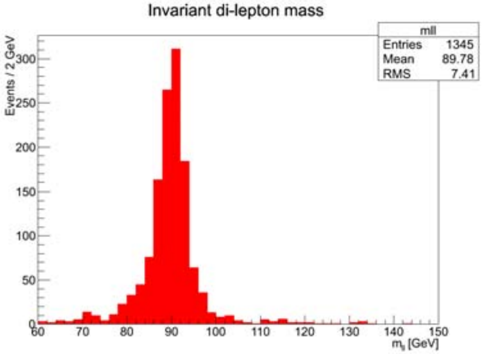
And it works, no problem!
plots on following slides

Plots Physics Analysis (2)

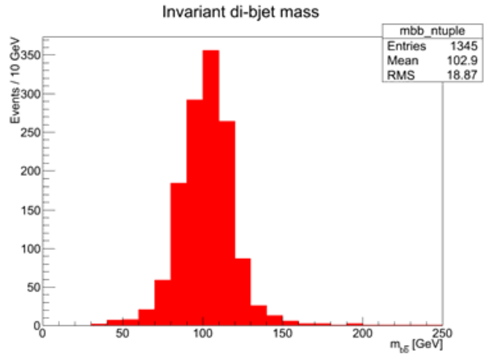
HZ→bbll sample

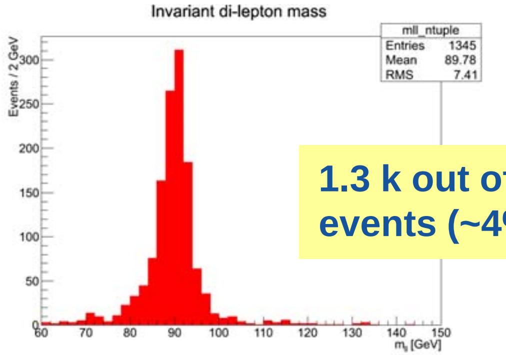
Database analysis





Ntuple analysis



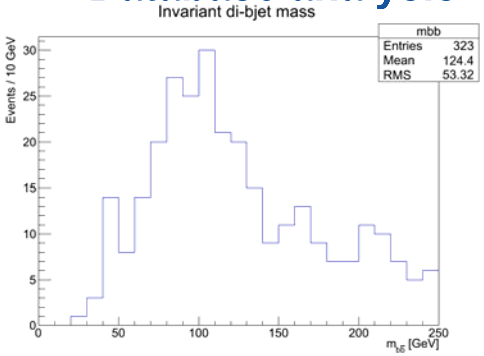


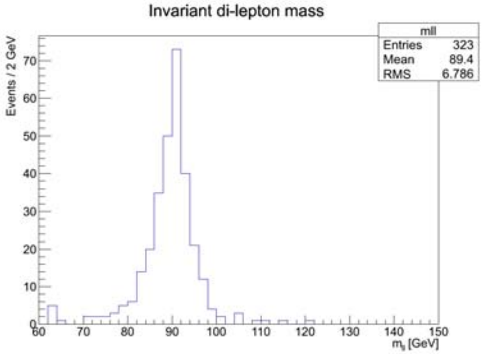
1.3 k out of 30 k events (~4%)

Plots Physics Analysis (2)

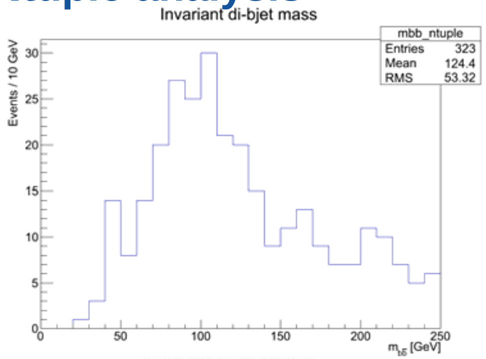
Z→ll+2/3/4 jets sample

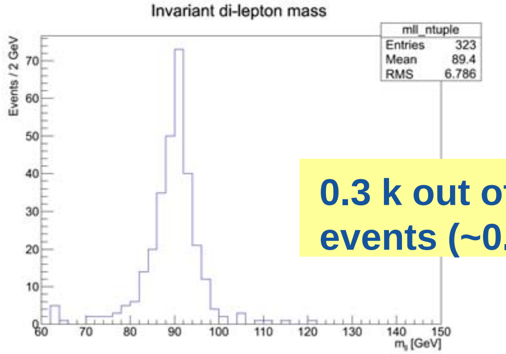
Database analysis





Ntuple analysis





0.3 k out of 1662 k events (~0.02%)

Timing Physics Analysis (2)

ZH→llbb sample:	mv1Eval (external)	fl_w_Comb>1.55
Ntuple analysis:	**15** s	**12** s
Database analysis:	**21** s	**18** s
Z→ll + jets sample:		
Ntuple analysis:	**549** s	**508** s
Database analysis:	**583** s	**333** s

The database analysis lost a lot of time by adding the use of a function from an external C library!

To test the cause of this delay, I created a test-query that only does the "jet"-part of the analysis and which separates the mv1Eval selection

Test-query: Jet-only selection

```
with sel_jet as (select /*+ MATERIALIZE FULL("jet") */ "jet_i","EventNo_RunNo","E","pt","phi","eta","fl_w_IP3D",
"fl_w_SV1", "fl_w_JetFitterCOMBNN" from "jet" where "pt">25000. and abs("eta")<2.5),
sel_bjet as (select /*+ MATERIALIZE */ "jet_i","EventNo_RunNo","E","pt","phi","eta" from sel_jet where
MV1.mv1Eval_fromExternal("fl_w_IP3D","fl_w_SV1","fl_w_JetFitterCOMBNN","pt","eta")>0.60173),
sel_jet_count as (select "EventNo_RunNo" from sel_bjet group by "EventNo_RunNo" HAVING MAX("pt")>45000. and
COUNT(*) = 2)
select "EventNo_RunNo",
PHYSANALYSIS.INV_MASS_JETS(jet0."E",jet1."E",jet0."pt",jet1."pt",jet0."phi",jet1."phi",jet0."eta",jet1."eta")/1000. as
"DiJetMass" from sel_jet_count INNER JOIN sel_jet jet0 USING ("EventNo_RunNo") INNER JOIN sel_jet jet1 USING
("EventNo_RunNo") where jet0."jet_i"<jet1."jet_i" and
PHYSANALYSIS.pass_bjet_pair_selection(jet0."pt"/1000.,jet1."pt"/1000.,jet0."phi",jet1."phi",jet0."eta",jet1."eta") = 1 ;
```

The query separates the jet-selection into two parts, the second part calls the external function

SQL monitoring
Di-jet-mass select Z→ll+jets

Monitored SQL Execution Details

Overview

SQL ID b8qumpzrjp026
Execution Started Thu Nov 15, 2012 5:33:30 PM
Last Refresh Time Thu Nov 15, 2012 5:42:01 PM
Execution ID 33554432
User NTUP_TOP_ZNP0_5
Fetch Calls 202

Time & Wait Statistics

Duration 8.5m
Database Time 6.6m
PL/SQL & Java 3.4s
Wait Activity % 100

IO Statistics

Buffer Gets 6,168K
IO Requests 49K
IO Bytes 46GB

Details

Plan Statistics | Plan | Activity | Metrics

Plan Hash Value 753856098

TIP: Right mouse click on the table allows to toggle between IO Requests and IO Bytes

Operation	Name	Estimated...	Cost	Timeline(511s)	Execu...	Actual ...	Memor...	Temp (...	IO Requests	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	20K				2.25	
TEMP TABLE TRANSFORMATION					1	20K					
LOAD AS SELECT					1	1	529KB		1,701	6.74	1.49
TABLE ACCESS FULL	jet	266K	1,613K		1	4,317K			46K	38	90
LOAD AS SELECT					1	1	529KB		26		
VIEW		266K	896							46	.75
TABLE ACCESS FULL	SYS_TEMP_0FD9FCD7	266K	896		1	4,317K			432	2.25	8.21
HASH JOIN		1	2,960		1	20K	2MB			3.37	
HASH JOIN		133	2,062		1	12K	1MB			1.12	
VIEW		133	1,163		1	2,921					
FILTER					1	2,921					
HASH GROUP BY		133	1,163		1	97K	3MB	3MB	40		
VIEW		266K	1,149		1	101K					
TABLE ACCESS FULL	SYS_TEMP_0FD9FCD7	266K	1,149		1	101K			7		
VIEW		266K	896		1	4,317K					
TABLE ACCESS FULL	SYS_TEMP_0FD9FCD7	266K	896		1	4,317K					
VIEW		266K	896		1	4,317K					
TABLE ACCESS FULL	SYS_TEMP_0FD9FCD7	266K	896		1	4,317K					

First Active: 163s; Duration: 343s

160 s for the full table scan
343 s for mv1Eval_fromExternal

External library functions continued

Note: if I replace the MV1-algorithm with a function that does “return 1.” the time to process all the rows is still >300 seconds

The “mv1Eval”-function is being called for every row via the external procedure agent (“extproc”)

The agents runs in its own private address space and exchanges input/output parameters between the oracle process and the external library code using IPC

The IPC overhead is (far) higher than the actual cost of the calculation!

Solution is using Java!

Java provides a controlled environment executed within the same process and address space as the oracle process

I’m still too lazy/stupid to rewrite the C++ algorithm in Java…

So I tried to call my C++ library using JNI from Java !

PL/SQL calling Java calling C++

PL/SQL

```
FUNCTION mv1Eval_java( w_IP3D IN NUMBER, w_SV1 IN NUMBER, w_JetFitterCombNN IN NUMBER,
jet_pt IN NUMBER, jet_eta IN NUMBER ) return double precision
as language java
name 'MV1_interface.mv1Eval(double, double,double,double,double) return double';
```

Java

```
public class MV1_interface {
   public native static double mv1Eval(double fl_w_IP3D, double fl_w_SV1, double fl_w_JetFitterCOMBNN, double pt, double eta);
   static{ System.loadLibrary("MV1_interface.so");}  }
```

C-interface calling C++

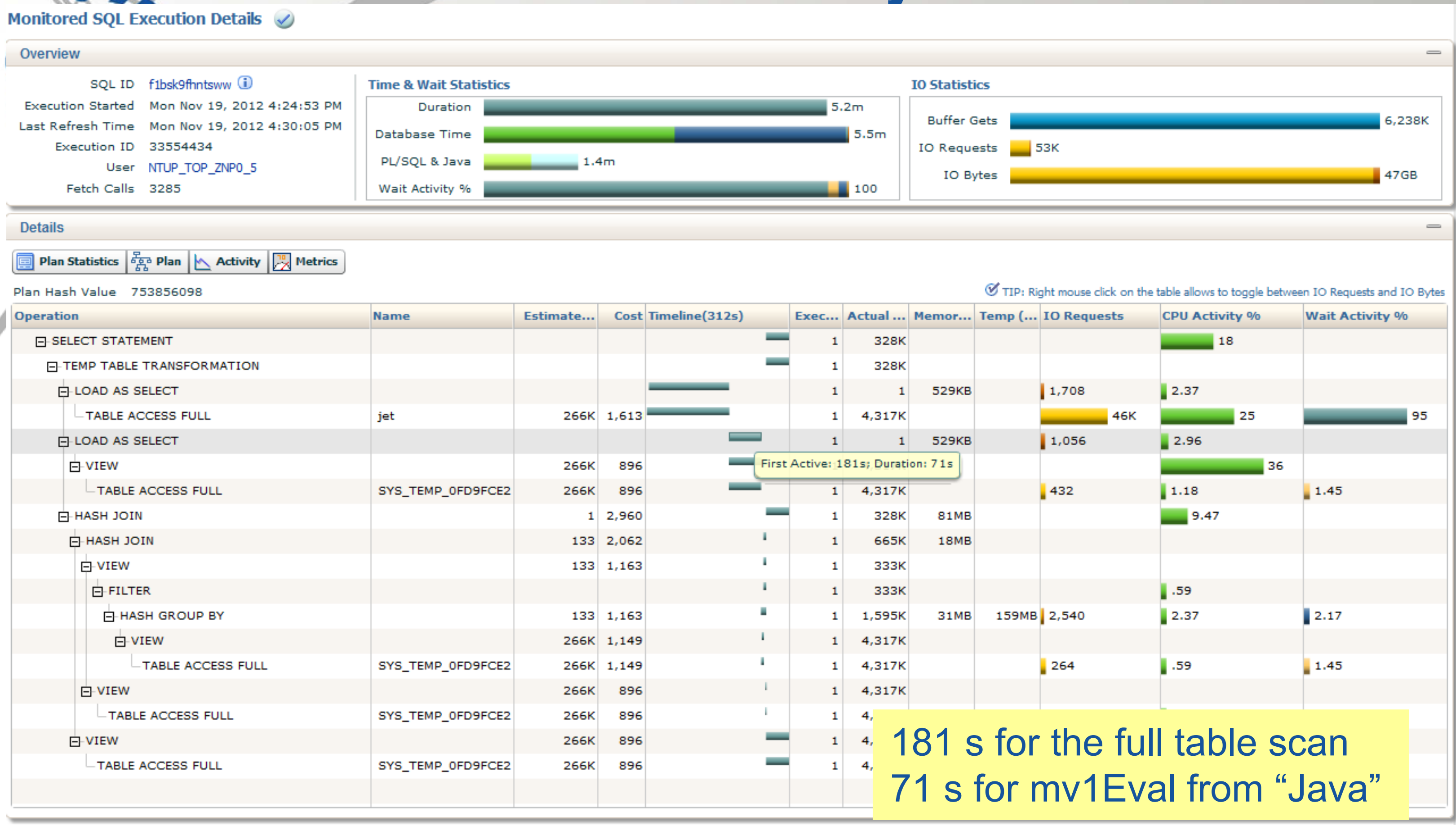
```
JNIEXPORT jdouble JNICALL Java_MV1_1interface_mv1Eval
(JNIEnv *, jclass, jdouble w_IP3D, jdouble w_SV1, jdouble w_JetFitterCombNN, jdouble jet_pt, jdouble jet_eta){
  double value = mv1Eval(w_IP3D, w_SV1, w_JetFitterCombNN, jet_pt, jet_eta);
  return value; }
```

Set permission to load library!

```
exec dbms_java.grant_permission('MLIMPER','SYS:java.lang.RuntimePermission','loadLibrary.MV1_interface.so','');
```

And it works! All the (pre-selected) rows of the “jet”-table are processed in 70 seconds instead >300 seconds

SQL monitoring
Di-jet-mass select Z→ll+jets



Monitored SQL Execution Details

Overview

SQL ID	f1bsk9fhntsww
Execution Started	Mon Nov 19, 2012 4:24:53 PM
Last Refresh Time	Mon Nov 19, 2012 4:30:05 PM
Execution ID	33554434
User	NTUP_TOP_ZNP0_5
Fetch Calls	3285

Time & Wait Statistics

Duration	5.2m
Database Time	5.5m
PL/SQL & Java	1.4m
Wait Activity %	100

IO Statistics

Buffer Gets	6,238K
IO Requests	53K
IO Bytes	47GB

Details

Plan Statistics | Plan | Activity | Metrics

Plan Hash Value 753856098

TIP: Right mouse click on the table allows to toggle between IO Requests and IO Bytes

Operation	Name	Estimate...	Cost	Timeline(312s)	Exec...	Actual ...	Memor...	Temp (...	IO Requests	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	328K				18	
TEMP TABLE TRANSFORMATION					1	328K					
LOAD AS SELECT					1	1	529KB		1,708	2.37	
TABLE ACCESS FULL	jet	266K	1,613		1	4,317K			46K	25	95
LOAD AS SELECT					1	1	529KB		1,056	2.96	
VIEW		266K	896	First Active: 181s; Duration: 71s						36	
TABLE ACCESS FULL	SYS_TEMP_0FD9FCE2	266K	896		1	4,317K			432	1.18	1.45
HASH JOIN		1	2,960		1	328K	81MB			9.47	
HASH JOIN		133	2,062		1	665K	18MB				
VIEW		133	1,163		1	333K					
FILTER					1	333K				.59	
HASH GROUP BY		133	1,163		1	1,595K	31MB	159MB	2,540	2.37	2.17
VIEW		266K	1,149		1	4,317K					
TABLE ACCESS FULL	SYS_TEMP_0FD9FCE2	266K	1,149		1	4,317K			264	.59	1.45
VIEW		266K	896		1	4,317K					
TABLE ACCESS FULL	SYS_TEMP_0FD9FCE2	266K	896		1	4,					
VIEW		266K	896		1	4,					
TABLE ACCESS FULL	SYS_TEMP_0FD9FCE2	266K	896		1	4,					

181 s for the full table scan
71 s for mv1Eval from "Java"

Timing Physics Analysis (2)

ZH→llbb sample:	mv1Eval_java	mv1Eval (external)	fl_w_Comb>1.55
Ntuple analysis:	**15** s	15 s	12 s
Database analysis:	**19** s	21 s	18 s
Z→ll + jets sample:			
Ntuple analysis:	**549** s	549 s	508 s
Database analysis:	**359** s	583 s	333 s

Now running on the Z+jets sample from the database is faster again!

Finally I'll show how I tried to improve the DB performance by changing my query:

- pre-select events passing the jet-pair criteria
- access the other tables using the index on EventNo_RunNo, so that only those rows that passed the jet-criteria have to be processed

SQL using index scan after jet-select (part 1)

```
with sel_jet as (select /*+ MATERIALIZE FULL("jet") */ "jet_i","EventNo_RunNo","E","pt","phi","eta" from "jet"  where "pt">25000. and abs("eta")<2.5 and MV1.mv1Eval_java("fl_w_IP3D","fl_w_SV1","fl_w_JetFitterCOMBNN","pt","eta")>0.60173 ),
sel_jet_count as (select "EventNo_RunNo" from sel_jet group by "EventNo_RunNo" HAVING MAX("pt")>45000. and COUNT(*) = 2),
sel_jet_events as (select /*+ MATERIALIZE */ "EventNo_RunNo",PHYSANALYSIS.INV_MASS_JETS(jet0."E",jet1."E",jet0."pt",jet1."pt",jet0."phi",jet1."phi",jet0."eta",jet1."eta")/1000. as "DiJetMass“ from sel_jet_count INNER JOIN sel_jet jet0 USING ("EventNo_RunNo") INNER JOIN sel_jet jet1 USING ("EventNo_RunNo") where jet0."jet_i"<jet1."jet_i" and PHYSANALYSIS.pass_bjet_pair_selection(jet0."pt"/1000.,jet1."pt"/1000.,jet0."phi",jet1."phi",jet0."eta",jet1."eta") = 1),
sel_electron as (select /*+ MATERIALIZE */ "electron_i","EventNo_RunNo","E","px","py","pz" from "electron" INNER JOIN sel_jet_events USING ("EventNo_RunNo") where PHYSANALYSIS.IS_ELECTRON("pt","eta","author","mediumWithTrack", 20000., 2.5) = 1 ),
sel_electron_count as (select "EventNo_RunNo",COUNT(*) as "el_sel_n" from sel_electron group by "EventNo_RunNo"),
sel_muon as (select /*+ MATERIALIZE */  "muon_i","EventNo_RunNo","E","px","py","pz" from "muon" INNER JOIN sel_jet_events USING ("EventNo_RunNo") where PHYSANALYSIS.IS_MUON("muon_i", "pt", "eta", "phi", "E", "me_qoverp_exPV", "id_qoverp_exPV","me_theta_exPV", "id_theta_exPV", "id_theta", "isCombinedMuon", "isLowPtReconstructedMuon","tight","expectBLayerHit", "nBLHits", "nPixHits","nPixelDeadSensors", "nPixHoles", "nSCTHits","nSCTDeadSensors", "nSCTHoles", "nTRTHits", "nTRTOutliers",0,20000.,2.4) = 1 ),
sel_muon_count as (select "EventNo_RunNo",COUNT(*) as "mu_sel_n" from sel_muon group by "EventNo_RunNo" ),
```

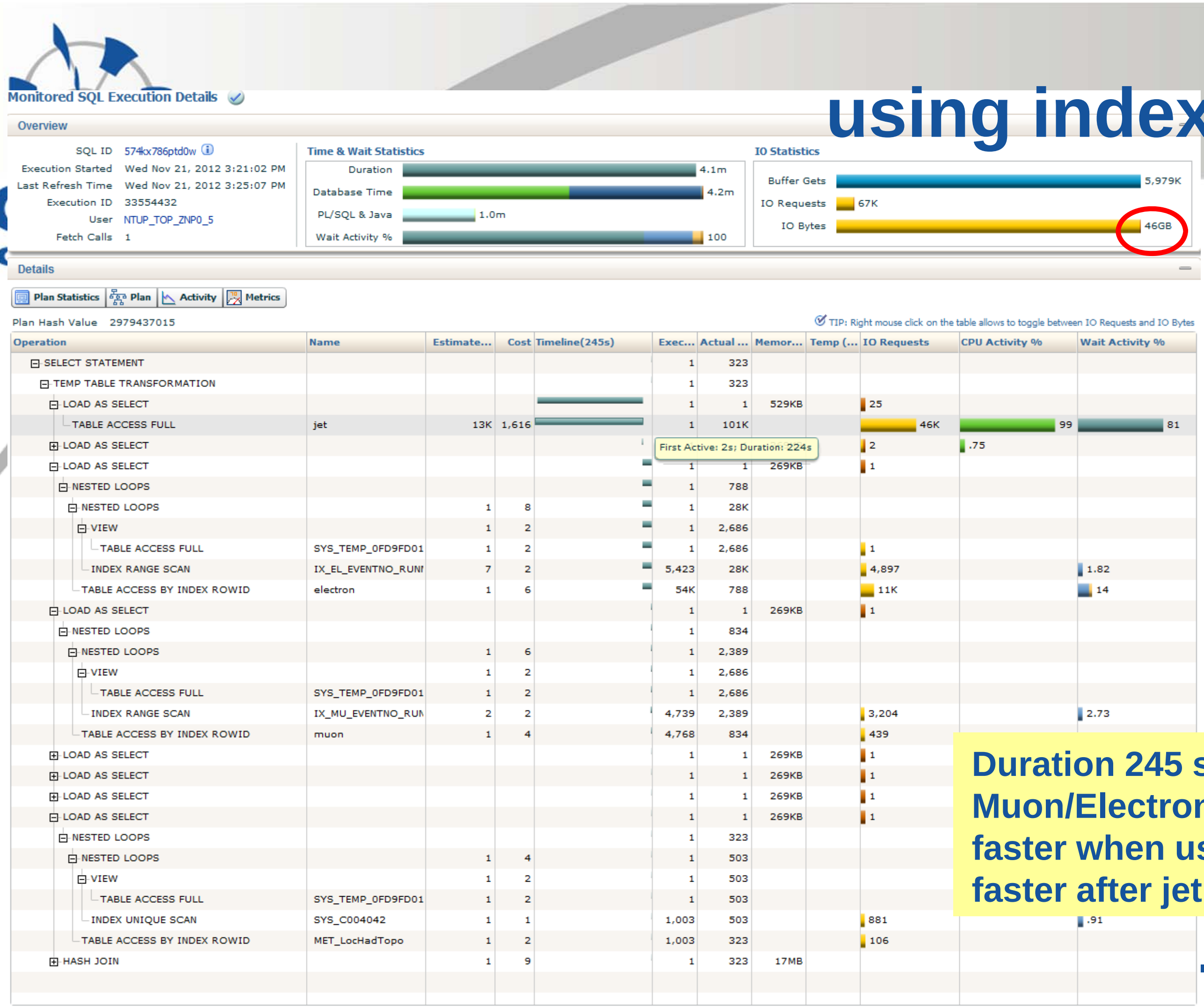
Query same as before, but removed FULL table scan hints for electron, muon and MET selection (and jet-selection first)

SQL using index scan after jet-select (part 2)

sel_mu_el_events as (select /*+ MATERIALIZE */ "EventNo_RunNo","el_sel_n","mu_sel_n" from sel_jet_events LEFT OUTER JOIN sel_electron_count USING ("EventNo_RunNo") LEFT OUTER JOIN sel_muon_count USING ("EventNo_RunNo") where ("el_sel_n"=2 and "mu_sel_n" is NULL) or ("el_sel_n" is NULL and "mu_sel_n"=2)),
sel_electron_events as (select /*+ MATERIALIZE */
"EventNo_RunNo",PHYSANALYSIS.INV_MASS_LEPTONS(el0."E",el1."E",el0."px",el1."px",el0."py",el1."py",el0."pz",el1."pz")/1000. as "DiElectronMass“ from sel_mu_el_events INNER JOIN sel_electron el0 USING ("EventNo_RunNo") INNER JOIN sel_electron el1 USING ("EventNo_RunNo") where el0."electron_i"<el1."electron_i"),
sel_muon_events as (select /*+ MATERIALIZE */
"EventNo_RunNo",PHYSANALYSIS.INV_MASS_LEPTONS(muon0."E",muon1."E",muon0."px",muon1."px",muon0."py",muon1."py",muon0."pz",muon1."pz")/1000. as "DiMuonMass"
from sel_mu_el_events INNER JOIN sel_muon muon0 USING ("EventNo_RunNo") INNER JOIN sel_muon muon1 USING ("EventNo_RunNo") where muon0."muon_i"<muon1."muon_i"),
sel_MET_events as (select /*+ MATERIALIZE */ "EventNo_RunNo","EventNumber","RunNumber" from "MET_LocHadTopo" INNER JOIN sel_mu_el_events USING ("EventNo_RunNo") where PHYSANALYSIS.pass_met_selection("etx","ety") = 1)
select "EventNo_RunNo","EventNumber","RunNumber",
"DiMuonMass","DiElectronMass","DiJetMass" from sel_muon_events FULL OUTER JOIN sel_electron_events USING ("EventNo_RunNo") INNER JOIN sel_jet_events USING ("EventNo_RunNo") INNER JOIN sel_MET_events USING ("EventNo_RunNo")

Query same as before, but removed FULL table scan hints for electron, muon and MET selection (and jet-selection first)

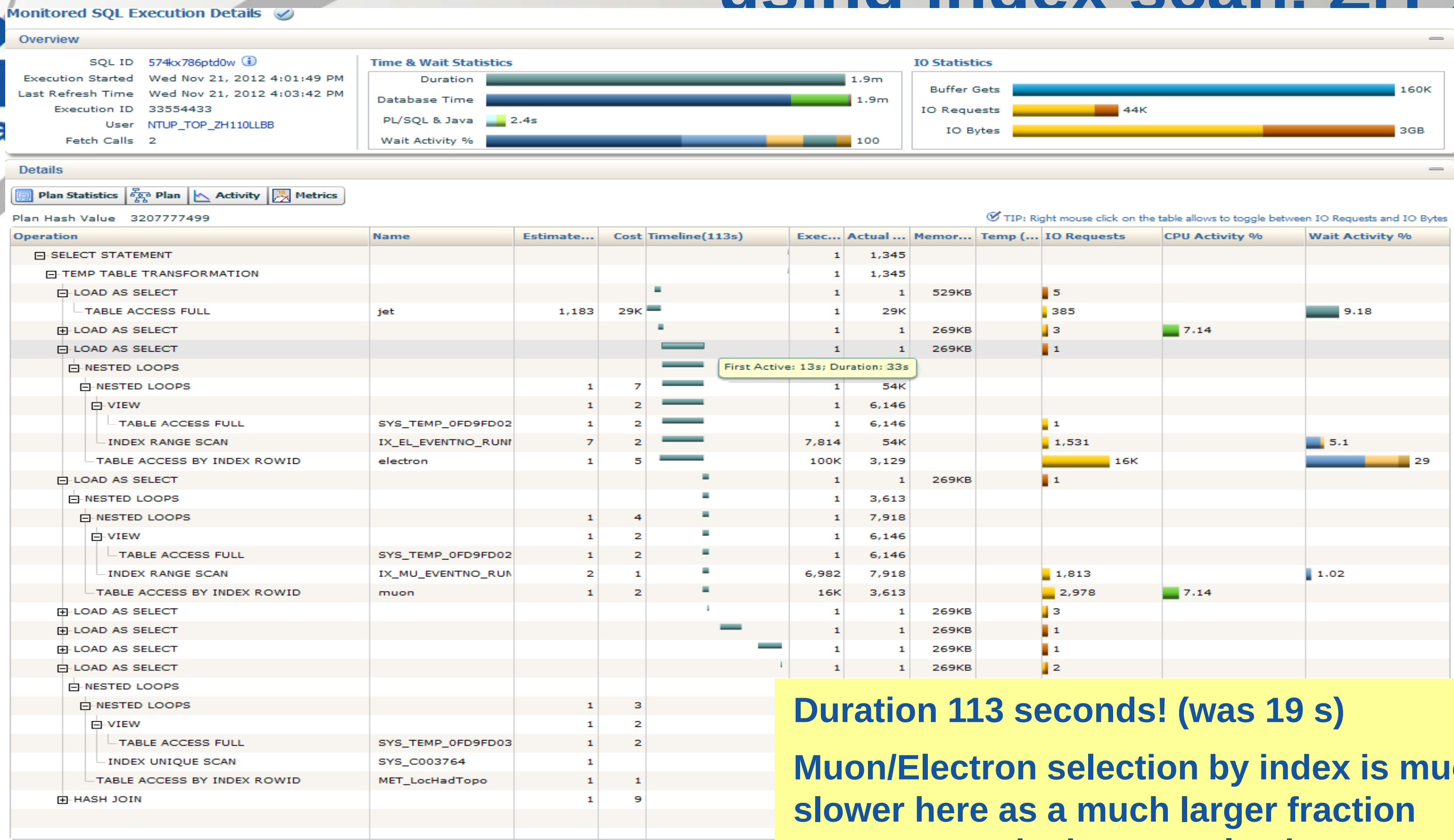
SQL monitoring using index scan, Z→ll+jets



Operation	Name	Estimate...	Cost	Timeline(245s)	Exec...	Actual ...	Memor...	Temp (...	IO Requests	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	323					
TEMP TABLE TRANSFORMATION					1	323					
LOAD AS SELECT					1	1	529KB		25		
TABLE ACCESS FULL	jet	13K	1,616		1	101K			46K	99	81
LOAD AS SELECT									2	.75	
LOAD AS SELECT					1	1	269KB		1		
NESTED LOOPS					1	788					
NESTED LOOPS		1	8		1	28K					
VIEW		1	2		1	2,686					
TABLE ACCESS FULL	SYS_TEMP_0FD9FD01	1	2		1	2,686			1		
INDEX RANGE SCAN	IX_EL_EVENTNO_RUNI	7	2		5,423	28K			4,897		1.82
TABLE ACCESS BY INDEX ROWID	electron	1	6		54K	788			11K		14
LOAD AS SELECT					1	1	269KB		1		
NESTED LOOPS					1	834					
NESTED LOOPS		1	6		1	2,389					
VIEW		1	2		1	2,686					
TABLE ACCESS FULL	SYS_TEMP_0FD9FD01	1	2		1	2,686					
INDEX RANGE SCAN	IX_MU_EVENTNO_RUN	2	2		4,739	2,389			3,204		2.73
TABLE ACCESS BY INDEX ROWID	muon	1	4		4,768	834			439		
LOAD AS SELECT					1	1	269KB		1		
LOAD AS SELECT					1	1	269KB		1		
LOAD AS SELECT					1	1	269KB		1		
LOAD AS SELECT					1	1	269KB		1		
NESTED LOOPS					1	323					
NESTED LOOPS		1	4		1	503					
VIEW		1	2		1	503					
TABLE ACCESS FULL	SYS_TEMP_0FD9FD01	1	2		1	503					
INDEX UNIQUE SCAN	SYS_C004042	1	1		1,003	503			881		.91
TABLE ACCESS BY INDEX ROWID	MET_LocHadTopo	1	2		1,003	323			106		
HASH JOIN		1	9		1	323	17MB				

Duration 245 seconds (was 359)
Muon/Electron selection significantly faster when using access by index
faster after jet pre-selection!

SQL monitoring using index scan, ZH→llbb



Monitored SQL Execution Details

Overview

SQL ID 574kx786ptd0w
Execution Started Wed Nov 21, 2012 4:01:49 PM
Last Refresh Time Wed Nov 21, 2012 4:03:42 PM
Execution ID 33554433
User NTUP_TOP_ZH110LLBB
Fetch Calls 2

Time & Wait Statistics

Duration 1.9m
Database Time 1.9m
PL/SQL & Java 2.4s
Wait Activity % 100

IO Statistics

Buffer Gets 160K
IO Requests 44K
IO Bytes 3GB

Details

Plan Statistics | Plan | Activity | Metrics

Plan Hash Value 3207777499

TIP: Right mouse click on the table allows to toggle between IO Requests and IO Bytes

Operation	Name	Estimate...	Cost	Timeline(113s)	Exec...	Actual ...	Memor...	Temp (...	IO Requests	CPU Activity %	Wait Activity %
SELECT STATEMENT					1	1,345					
TEMP TABLE TRANSFORMATION					1	1,345					
LOAD AS SELECT					1	1	529KB		5		
TABLE ACCESS FULL	jet	1,183	29K		1	29K			385		9.18
LOAD AS SELECT					1	1	269KB		3	7.14	
LOAD AS SELECT					1	1	269KB		1		
NESTED LOOPS				First Active: 13s; Duration: 33s							
NESTED LOOPS		1	7		1	54K					
VIEW		1	2		1	6,146					
TABLE ACCESS FULL	SYS_TEMP_0FD9FD02	1	2		1	6,146			1		
INDEX RANGE SCAN	IX_EL_EVENTNO_RUNI	7	2		7,814	54K			1,531		5.1
TABLE ACCESS BY INDEX ROWID	electron	1	5		100K	3,129			16K		29
LOAD AS SELECT					1	1	269KB		1		
NESTED LOOPS					1	3,613					
NESTED LOOPS		1	4		1	7,918					
VIEW		1	2		1	6,146					
TABLE ACCESS FULL	SYS_TEMP_0FD9FD02	1	2		1	6,146					
INDEX RANGE SCAN	IX_MU_EVENTNO_RUN	2	1		6,982	7,918			1,813		1.02
TABLE ACCESS BY INDEX ROWID	muon	1	2		16K	3,613			2,978	7.14	
LOAD AS SELECT					1	1	269KB		3		
LOAD AS SELECT					1	1	269KB		1		
LOAD AS SELECT					1	1	269KB		1		
LOAD AS SELECT					1	1	269KB		2		
NESTED LOOPS											
NESTED LOOPS		1	3								
VIEW		1	2								
TABLE ACCESS FULL	SYS_TEMP_0FD9FD03	1	2								
INDEX UNIQUE SCAN	SYS_C003764	1									
TABLE ACCESS BY INDEX ROWID	MET_LocHadTopo	1	1								
HASH JOIN		1	9								

Duration 113 seconds! (was 19 s)

Muon/Electron selection by index is much slower here as a much larger fraction events pass the jet-pre-selection

Timing Physics Analysis (2)

ZH→llbb sample:	mv1Eval_java	mv1Eval (external)	fl_w_Comb>1.55
Ntuple analysis:	**15** s	15 s	12 s
Database analysis, FULL:	**19** s	21 s	18 s
Database analysis, via index:	**113** s		
Z→ll + jets sample:			
Ntuple analysis:	**549** s	549 s	508 s
Database analysis, FULL:	**359** s	583 s	333 s
Database analysis, via index:	**247** s		

Best selection strategy depends on sample!

Note: I did not specify to use the index, rather I removed the hint forcing the full table scan, the query optimizer could have made a better decision for the ZH→llbb sample!

Timing with parallel execution

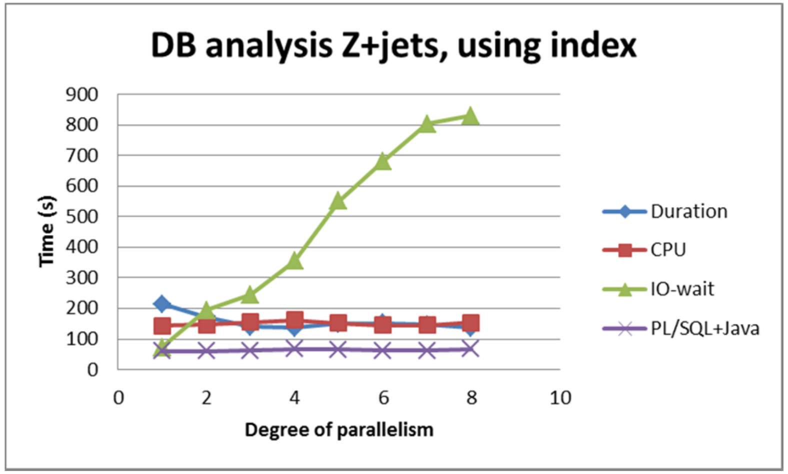
Repeat queries using "parallel X" hint on all tables (with m1Eval_java)

Parallelism brings the analysis times down to :

~210 s (full table scans)

~135 s (with index)

The IO-wait time is a bottle-neck



Timing with parallel execution

Repeat queries using "parallel X" hint on all tables (with m1Eval_java)

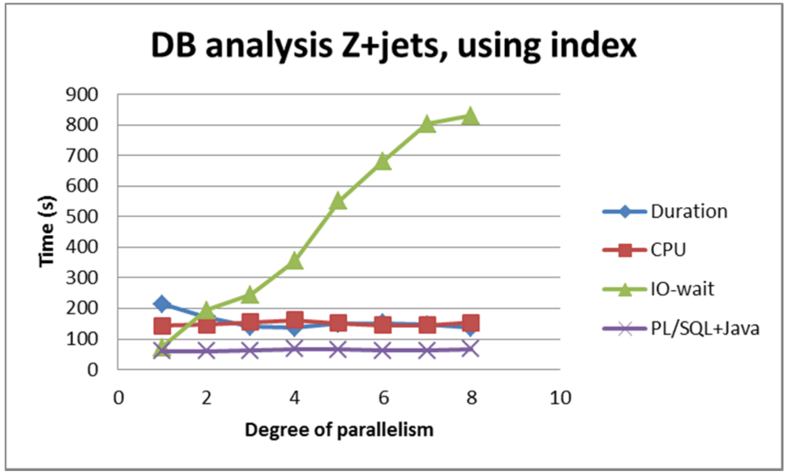
Parallelism brings the analysis times down to :

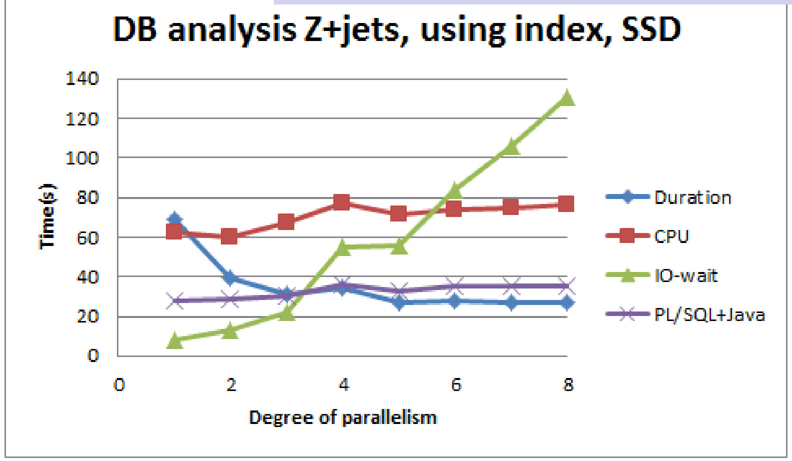
~210 s (full table scans)

~135 s (with index)

The IO-wait time is a bottle-neck

IO-wait reduced by moving data to SSD
Ntuple analysis on SSD only 62 s





Outlook

HZ→bbll (signal)

3 GB ntuple data

1.3 k out of 30 k events selected (~4%)

Analysis time*
DB: 19 s, ntuple: 15 s

Z→ll + jets (background)

170 GB ntuple data

0.3 k out of 1662 k events selected (~0.02%)

Analysis time*
DB: 247 s, ntuple: 549 s

2012 LHC data

60 TB ntuple data

? out of 1000 M events selected (<<0.01%)

**Database and ntuple-analysis run on same machine, timing fluctuation ~5%*

Still to do: analysis of real data, requires a much larger sample of events and will result in an even smaller percentage of events selected

I'd expect the time-gain with respect to the ntuple-analysis to be even greater

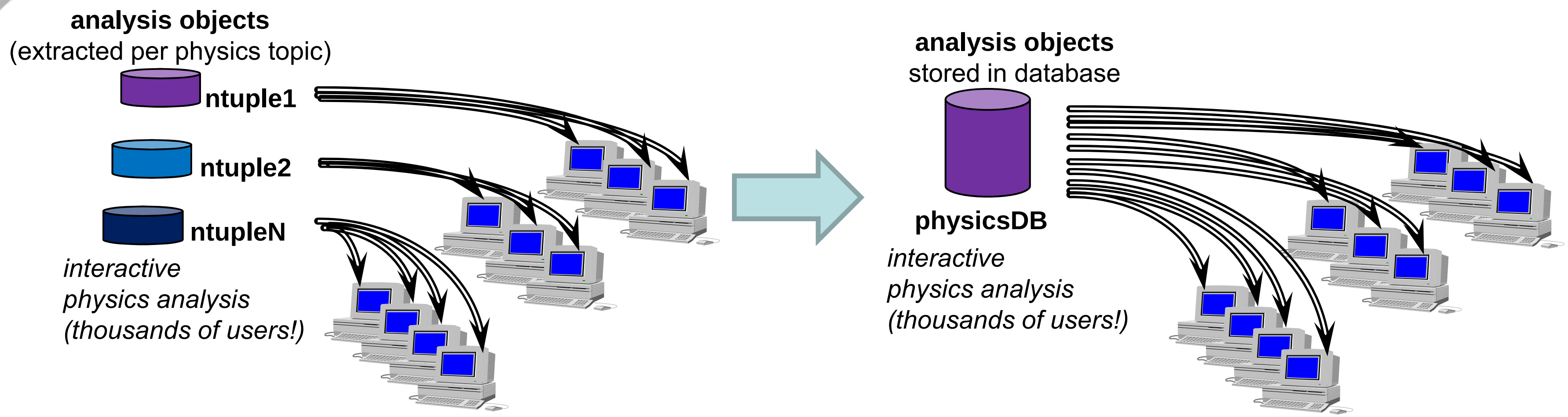
LHC Data in the database to be separated in subsets of data (per RunNumber)

- **analysis query to run simultaneous on each subset and histograms summed afterwards**

Outlook

Separate ntuple production for different physics topics but users still struggle to deal with the large data volumes (copy locally, filter, run on grid?)

The database would remove the need for separate ntuple production and can optimize CPU resources using parallel execution



How will the Oracle database handle a large number of users performing their own unique physics analysis studies at the same time?

Summary (1)

LHC data analysis in an Oracle database: a real “big data” challenge!

I study how to make this possible, but we are not implementing this just yet…

The database offers the advantage to store the data in a logical way and would remove the need for separate ntuple production for the different physics groups

Analysis is IO intensive, the challenge is to have a good performance while being able to cater to the requirements of all physics groups

Analysis code can be rewritten in SQL, but it is not trivial

Query optimizer can not estimate number of rows returned by a complicated selection, hints are generally required to optimize performance

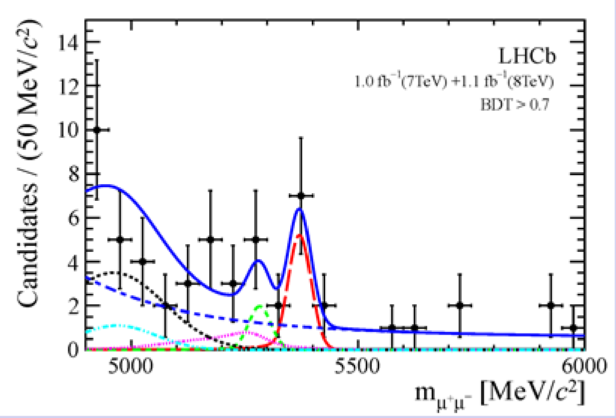
More complicated calculations might need to be done by external functions

Summary (2)

LHC physics analysis requires going through billions of events to find a handful of events that match the desired event topology

See for example the recently published B_s→ μμ result, billions of LHC collisions were processed to find an event-peak containing 10 events!

http://lhcb-public.web.cern.ch/lhcb-public/Welcome.html#BsMuMu3



Not as simple as pumping raw data from the LHC experiment straight into a database…

Raw data is reconstructed into analysis objects, reconstruction in the database too complicated (for now)

However, as shown today, the Oracle database can perform (basic) LHC physics analysis

Further studies needed to understand if the database can handle thousands of users accessing hundreds of petabytes of data *(and they all want their plots ASAP…)*

Want to know more about CERN and LHC?

Plenty more information available on-line! Here's a snapshot:

Animation of ATLAS proton collision event showing LHC acceleration chain:

- http://www.atlas.ch/multimedia/#di-jet-event

Study collision events with interactive CMS event displays:

- http://cms.web.cern.ch/content/cms-data-public

The first collisions in the LHCb's experiment – March 30th, 2010

- http://www.youtube.com/watch?v=0uoqrh51ZPY

Watch "ALICE Voyage inside the core of matter" at:

- http://aliceinfo.cern.ch/Public/Welcome.html