How to discover the Higgs Boson in an Oracle database

Maaike Limper

CERN openlab

Introduction

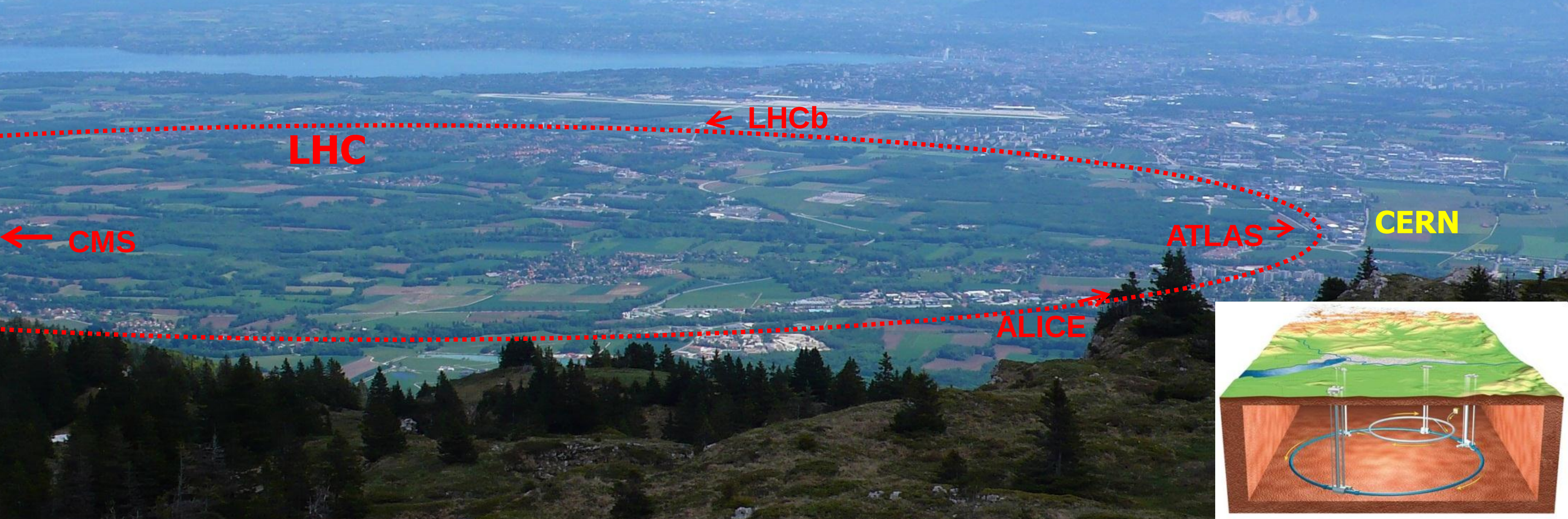
*“**CERN openlab** is a unique public-private partnership between CERN and leading ICT companies. Its mission is to accelerate the development of cutting-edge solutions to be used by the worldwide LHC community”* http://openlab.web.cern.ch

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

- Four main experiments recording events produced by the Large Hadron Collider: ATLAS, CMS, LHCb and ALICE
- Implementation of physics analysis in Oracle database based on my experience with the ATLAS experiment



Introduction

Some of the items I discuss today:

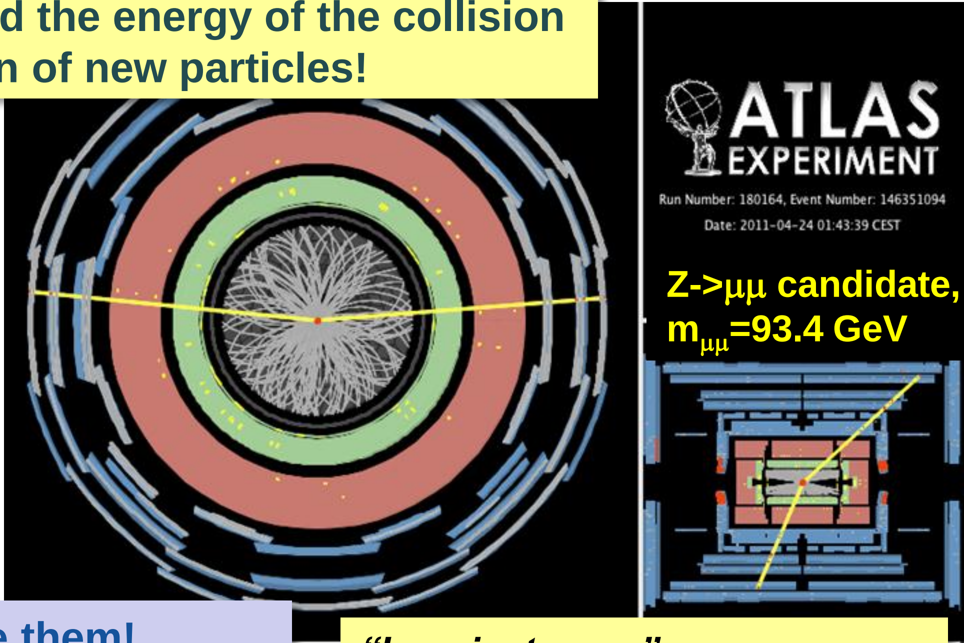
- **LHC physics analysis: how do we go from detector measurements to discovering new particles**
- **An example of a database structure containing analysis data**
- **An example of physics analysis code converted to SQL**
- **Using outside algorithms (C++/java) as part of the event selection**
- **Parallel execution**
- **Multiple Users**
- **Outlook**

***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*!**

Finding new particles…

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"

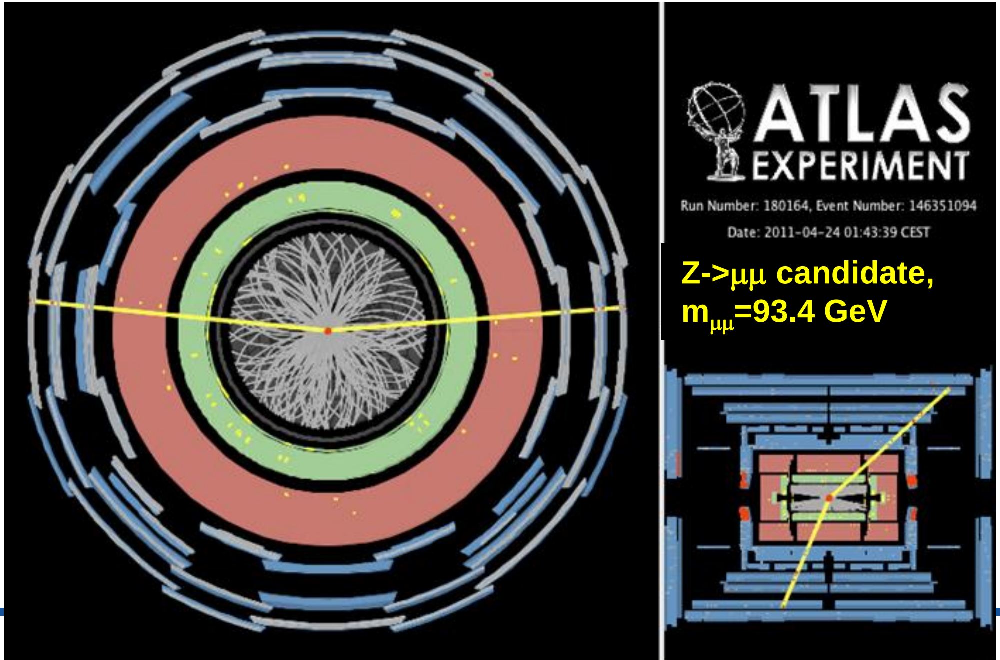
$$Mc^2 = (\sum E)^2 + \|\sum \vec{p}c\|^2$$

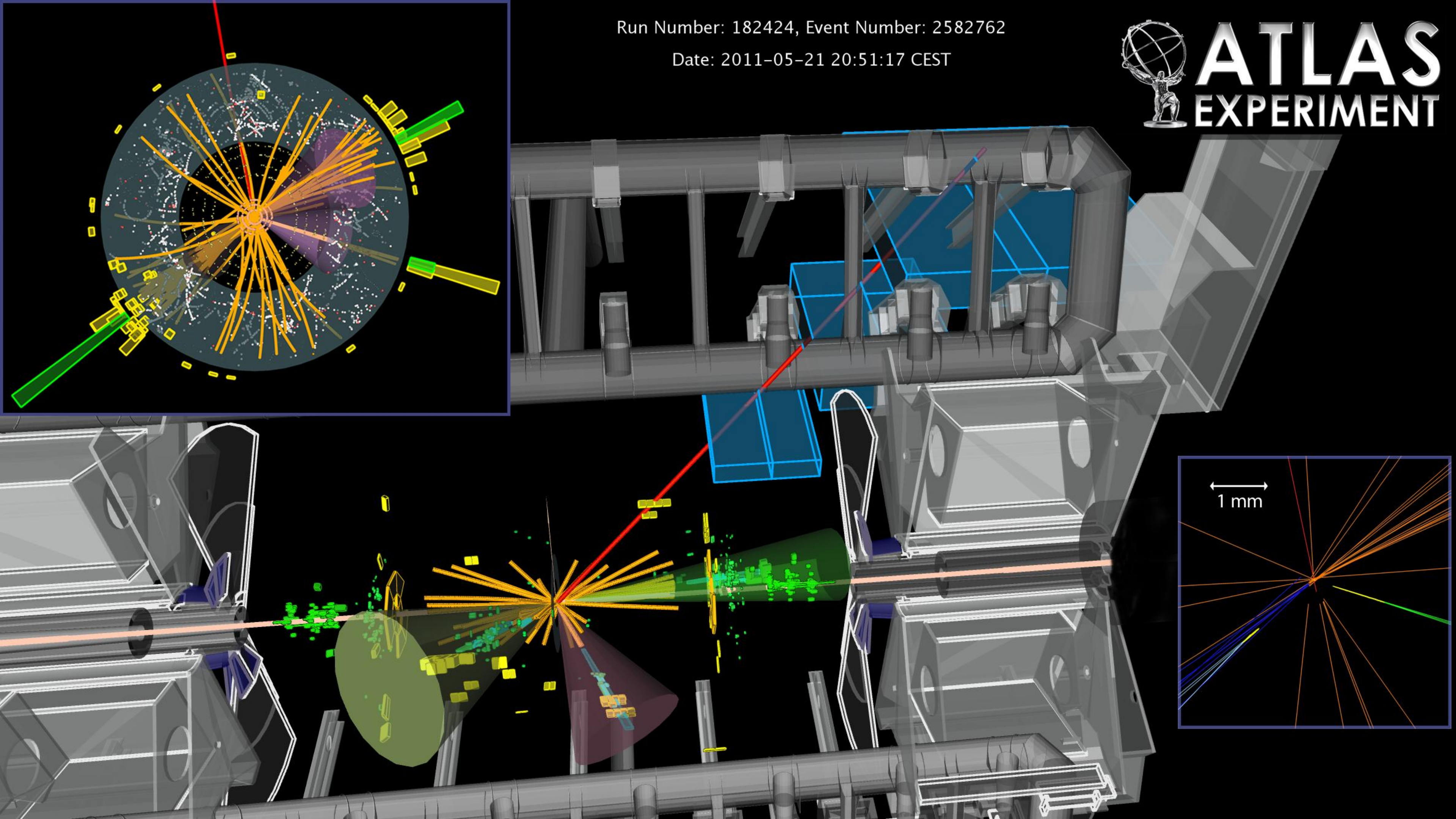
M=*invariant mass, equal to mass of decay particle*
$\sum E$=*sum of the energies of produced particles*
$\|\sum \vec{p}c\|$=*vector sum of momenta of produced particles*

Analysis versus reconstruction

Event Reconstruction focuses on creating physics objects from the information measured in the detector

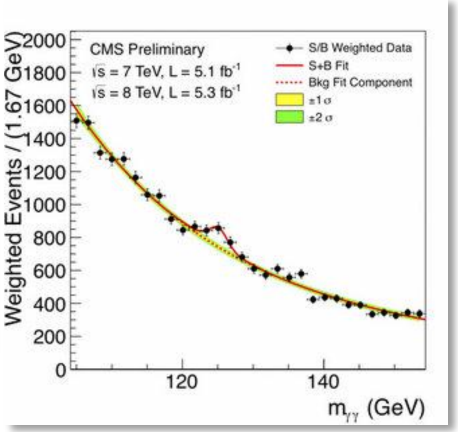
Event Analysis focuses on interpreting information from the reconstructed objects to determine what type of event took place

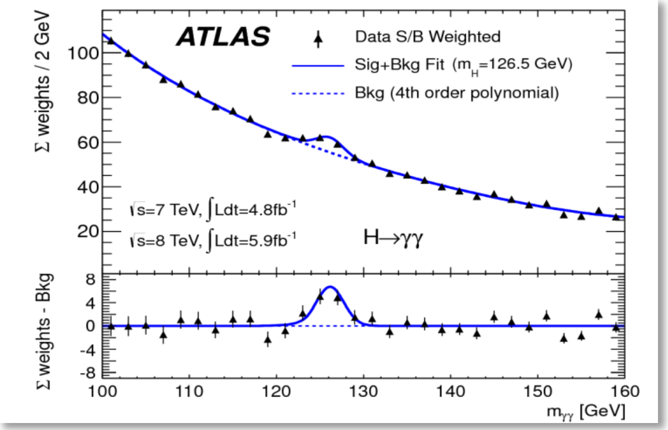



Run Number: 182424, Event Number: 2582762
Date: 2011-05-21 20:51:17 CEST
ATLAS
EXPERIMENT
1 mm

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





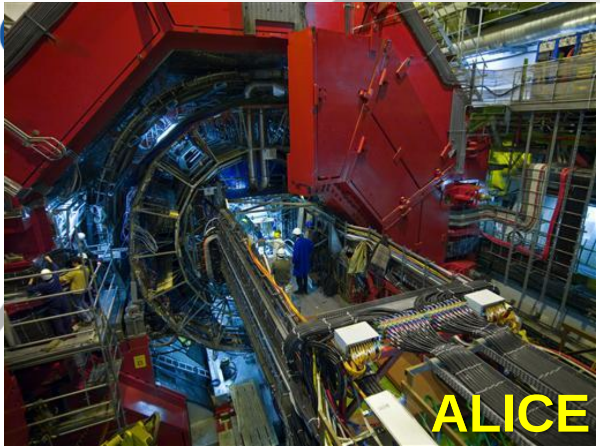
The discovery of a "Higgs boson-like" particle!

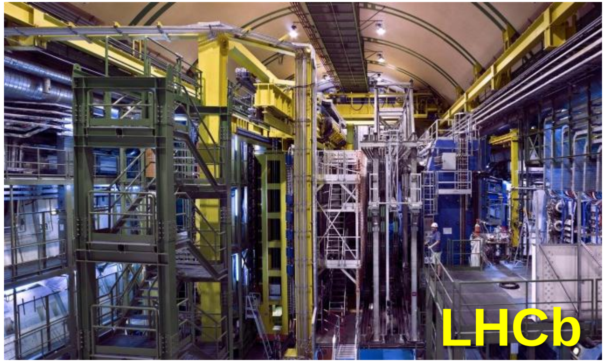
http://www.bbc.co.uk/news/world-18702455

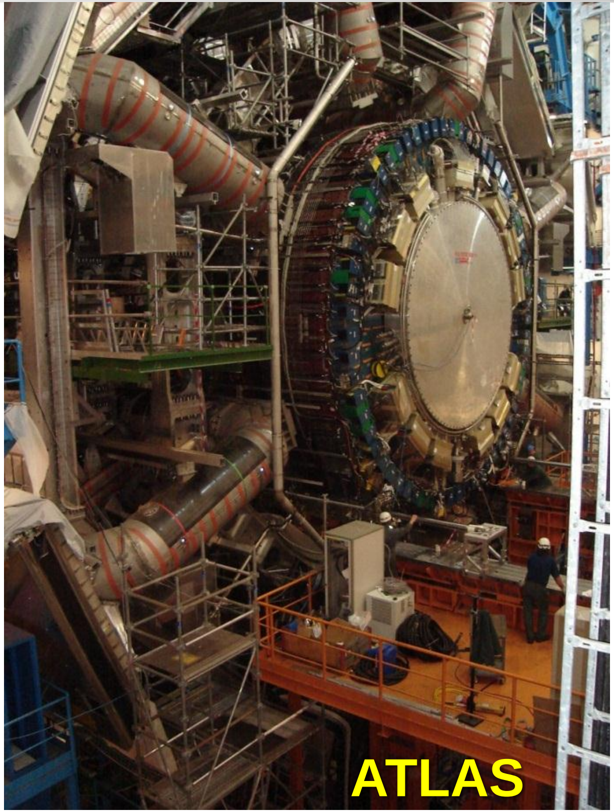
- The work of thousands of people!
- Operations of LHC and its experiments rely on databases for storing conditions data, log files etc.

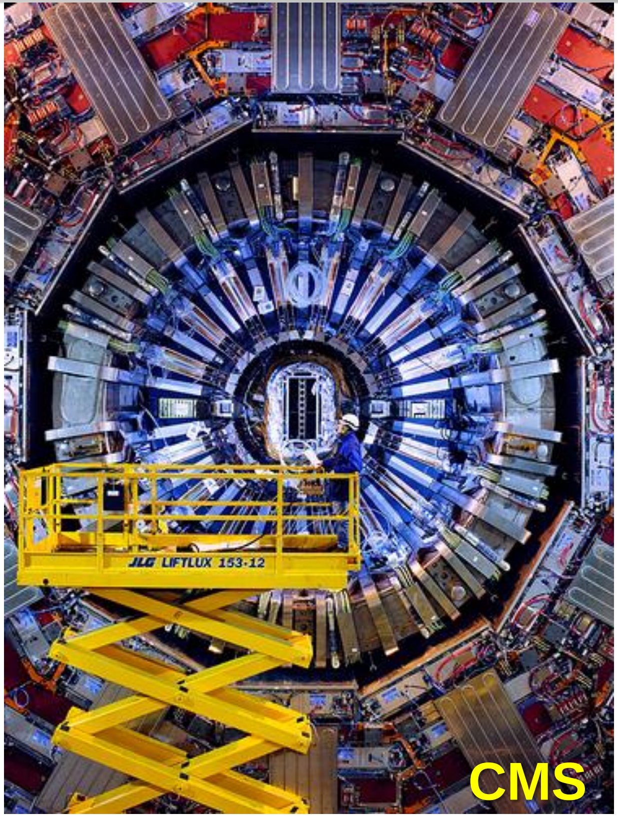
… but the data-points in these plots did not came out of a database !

Where does the data come from?

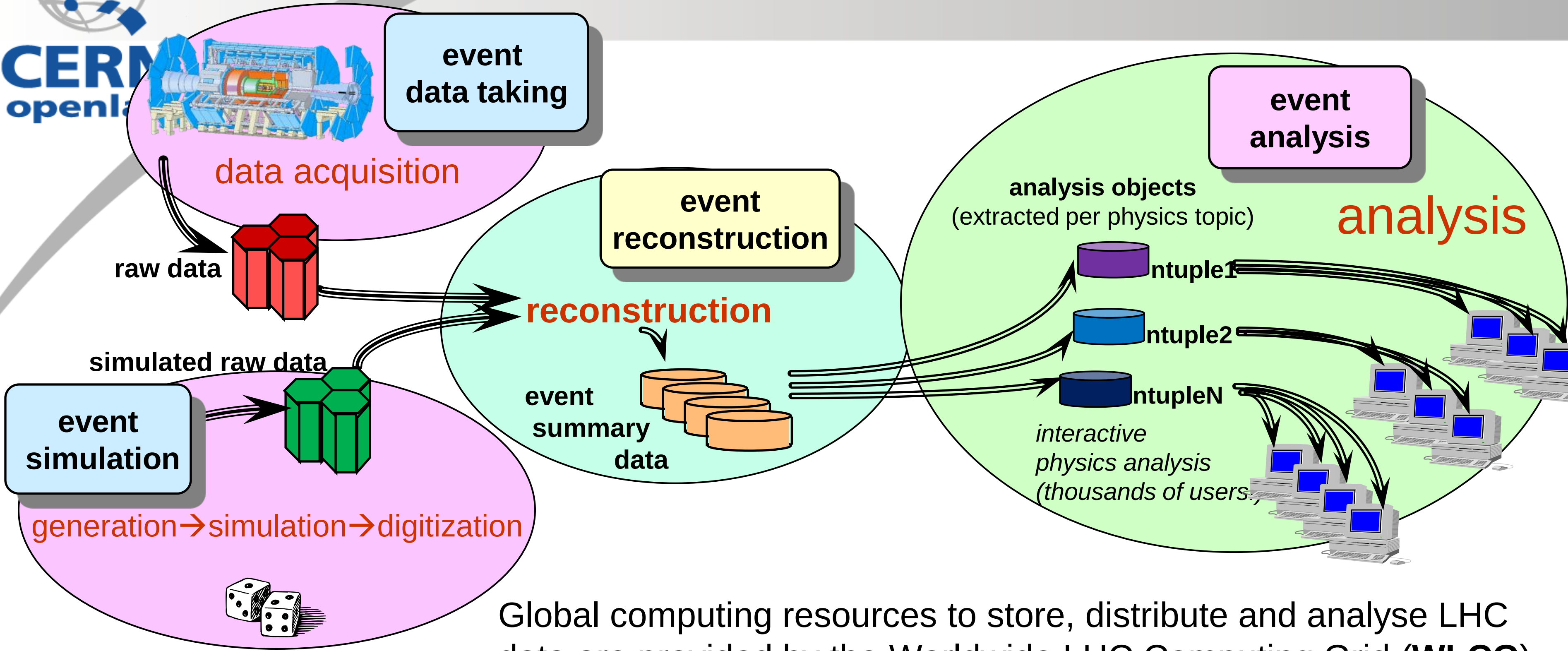








Where does the data come from?

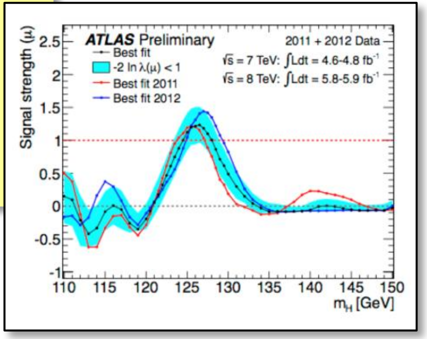


Global computing resources to store, distribute and analyse LHC data are provided by the Worldwide LHC Computing Grid (**WLCG**) which has more than 170 computing centres in 36 countries

Data analysis in practice

LHC Physics analysis is done with ROOT

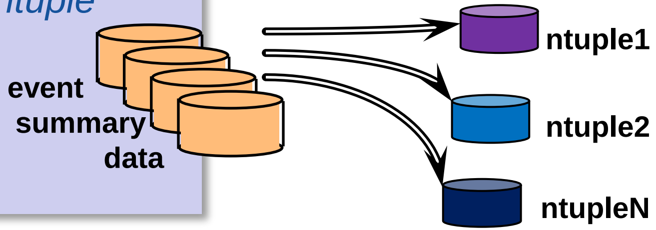
- Dedicated C++ framework developed by the High Energy Physics community, http://root.cern.ch
- Provides tools for plotting/fitting/statistic analysis etc.



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*

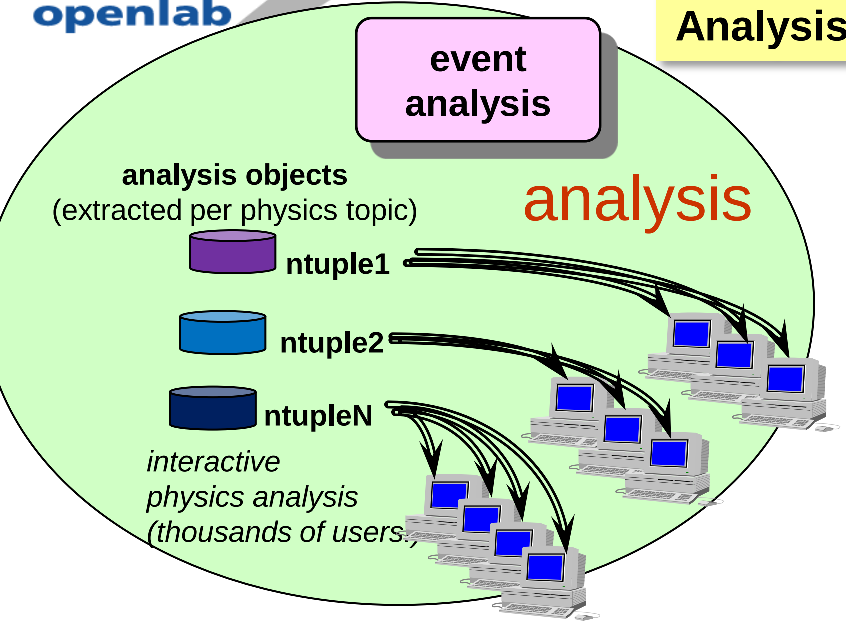


Data in ntuples is stored as a “TTree” object, with a “TBranch” for each variable

Optimized to reduce I/O: retrieving only TBranches neccessary for analyses from ntuple, data from loaded branches cached

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?

Physics 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

Physics analysis in a database

Database design philosophy:
Separate tables for different physics objects
Users read the object-tables relevant for their analysis
…and ignore the table that are not

Currently implemented 1042 variables, divided over 5 different tables

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

My test DB implementation contains ~75 GB of data
A real physics database containing all 2012 data would contain ~50 TB (“NTUP_TOP”-samples)

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 50 < GeV
- **electron selection:** require p_T>20 GeV and |η|<2.4, requirement on hits and holes on tracks, isolation criteria
- **muon selection:** require p_T>20 GeV and |η|<2.4, requirement on hits and holes on tracks, isolation criteria
- **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)

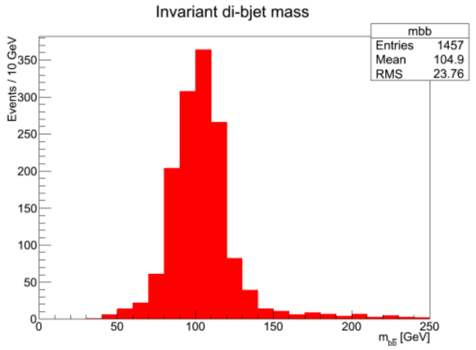
sel_electron_events as (select /*+ MATERIALIZE */ "EventNo_RunNo",PHYSANALYSIS.INV_MASS_LEPTONS(el0."E",el1."E",el0."px",el1."px",el0."py",el1."py",el0."pz",el1."pz")/1000. as "DiElectronMass" from sel_mu_el_events INNER JOIN sel_electron el0 USING ("EventNo_RunNo") INNER JOIN sel_electron el1 USING ("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")

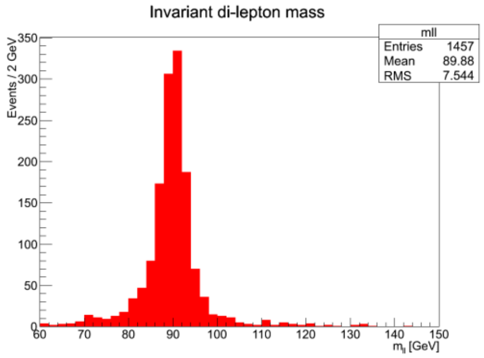
The final select-statement returns the invariant mass of the leptons and jets

Plots Physics Analysis (1)

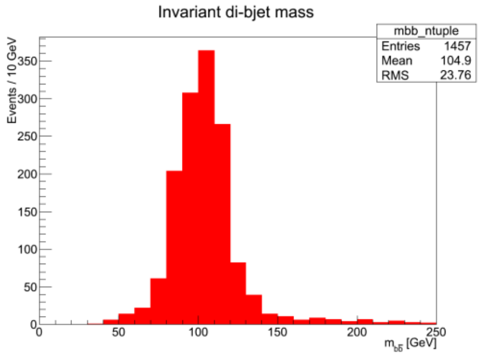
HZ→bbll sample

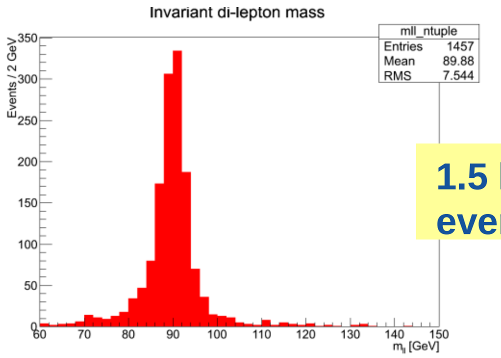
Database analysis





Ntuple analysis



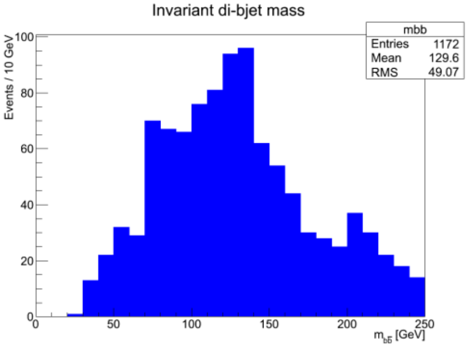


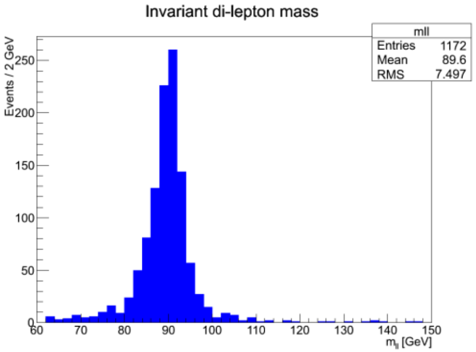
1.5 k out of 30 k events (~5%)

Plots Physics Analysis (1)

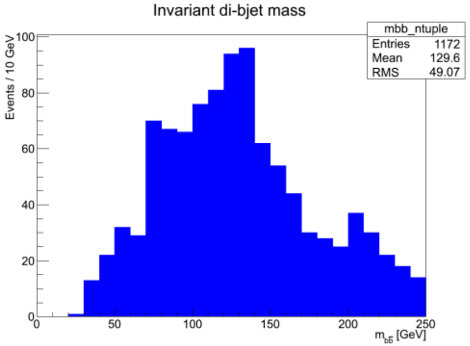
Z→ll+2/3/4 jets sample

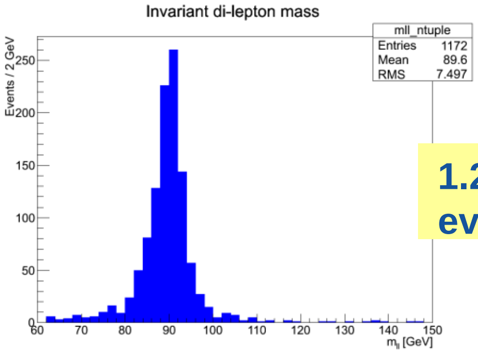
Database analysis





Ntuple analysis





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

Timing Physics Analysis (1)

Database runs on the same (itrac) machine as the root ntuple analysis

Ntuple-files and database-files use the same storage space (NFS)

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

ZH→llbb sample:

Ntuple analysis:	**12** seconds
Database analysis:	**18** seconds

Z→ll + jets sample:

Ntuple analysis:	**508** seconds
Database analysis:	**333** seconds

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				

Query time mainly due to full table scans

“MET”-table:	12 s
“electron”-table:	102 s
“muon”-table:	29 s
“jet”-table:	178 s

Physics Analysis (2)

What if a user can't (or does not want) to re-write a piece of more complicate analysis code in SQL?

Changed b-jet selection to re-calculate the jet "flavour weight", using some C++ code from ATLAS

"mv1Eval": a neural-network based algorithm that combines the output of different b-tagging weights to calculate an optimized b-tagging weight

Compile the code as a standalone library and you call it as an external function from SQL

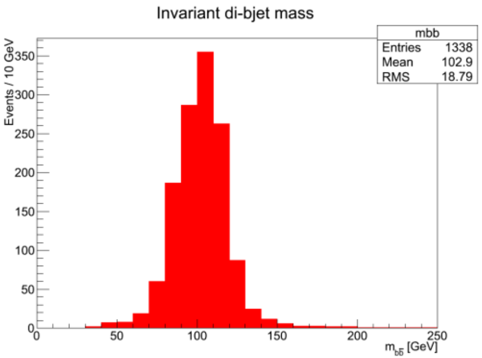
```
FUNCTION mv1Eval_fromExternal( w_IP3D double precision,  w_SV1 double precision, w_JetFitterCombNN double precision, jet_pt double precision, jet_eta double precision ) return double precision

 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);
```

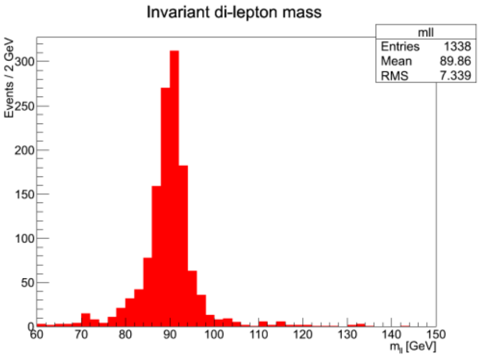
And it works, no problem!
plots on following slides

Plots Physics Analysis (2)

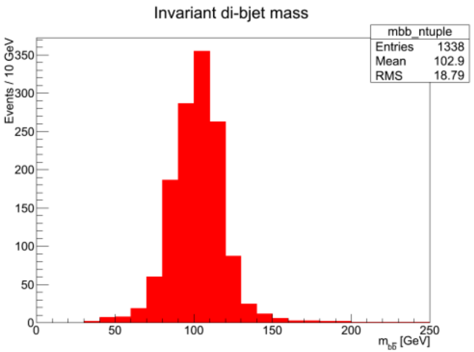
HZ→bbll sample

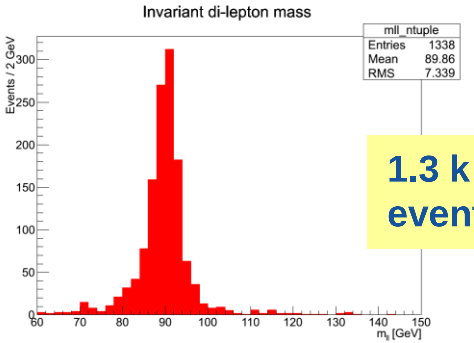
Database analysis





Ntuple analysis



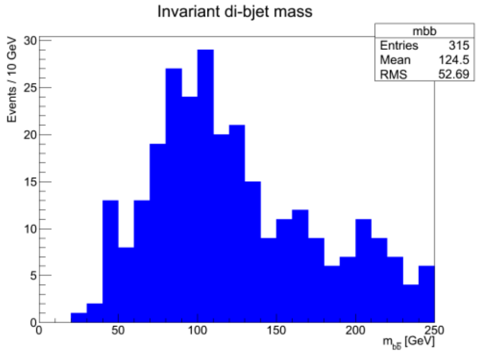


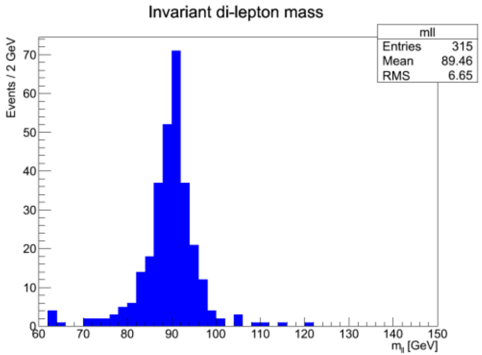
1.3 k out of 30 k events (~4%)

Plots Physics Analysis (2)

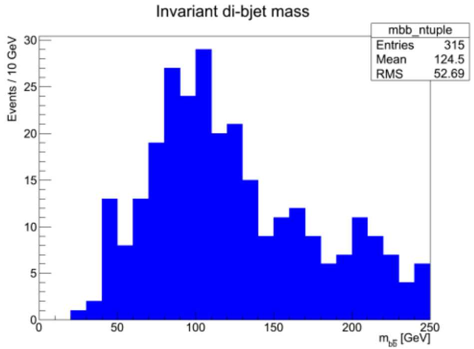
Z→ll+2/3/4 jets sample

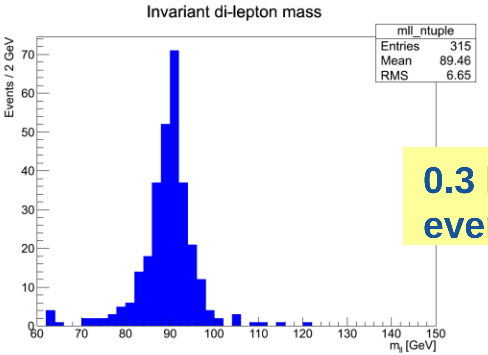
Database analysis





Ntuple analysis





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

Timing Physics Analysis (2)

ZH→llbb sample:	*fl_w_Comb>1.55*	***mv1Eval_C (external)***
Ntuple analysis:	12 s	**15** s
Database analysis:	18 s	**21** s
Z→ll + jets sample:		
Ntuple analysis:	508 s	**549 s**
Database analysis:	333 s	**583 s**

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

The SQL monitoring plan showed that the time spent on the full scan of the jet-table increased from 178 s to 428 s when using the external function

External library functions continued

When I replaced the MV1-algorithm with a function that only did "return 1."
the time to process all rows in the jet-table was still ~380 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

But I don't want to rewrite the code in Java…

So I tried to call my C++ library using Java Native Interface

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','');
```

Timing Physics Analysis (2)

ZH→llbb sample:	*fl_w_Comb>1.55*	*mv1Eval_C*	***mv1Eval_C_via_java***
Ntuple analysis:	12 s	15 s	**15** s
Database analysis:	18 s	21 s	**19** s
Z→ll + jets sample:			
Ntuple analysis:	508 s	549 s	**549** s
Database analysis:	333 s	583 s	**359** s

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 and "ptcone20"<0.1*"pt"),
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 and "ptcone20"<0.1*"pt"),
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)

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		

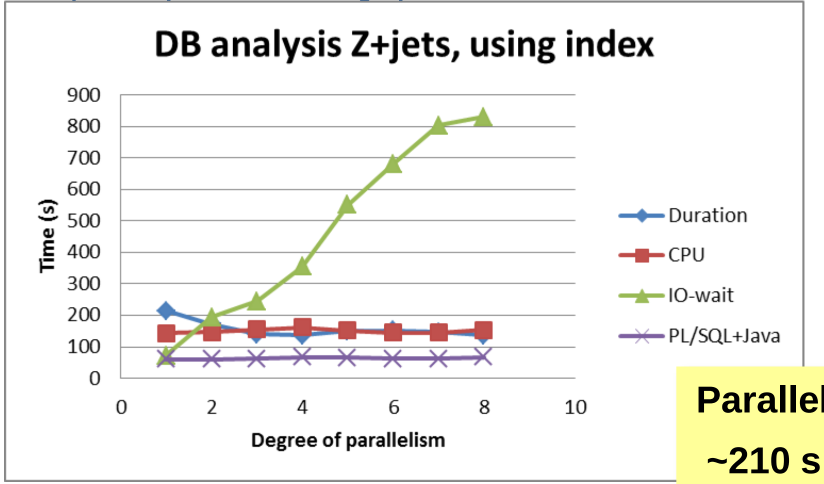
Best selection strategy depends on sample!

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!

Parallel execution

Test if analysis time can be reduced using parallel execution:

Repeat queries using "parallel X" on all tables:



** CPU,IO-wait and PL/SQL+Java time is sum of time over all parallel servers*

Parallelism brings the analysis times down to :

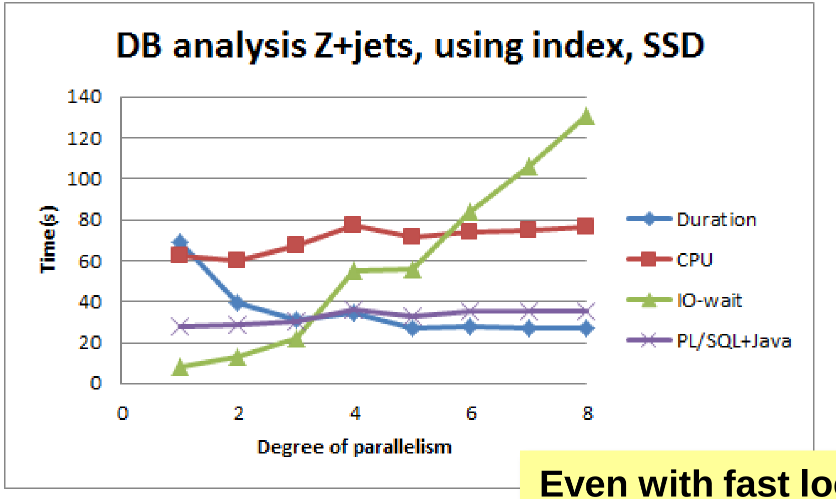
~210 s (full table scans)

~135 s (with index)

The IO-wait time is a bottle-neck preventing the parallelism from having a more significant effect

Parallel execution, with flash disk

Copied test setup to an improved setup to “devrac5”
more CPU power and <u>fast local flash disk storage</u>



Ntuple analysis: 62 s

Database analysis: 72 s (DOP=1)

Database analysis: 33 s (DOP=3)

Gain from parallelism higher on SSD but no more gain after DOP=3

Ntuples gain relatively more from move to SSD

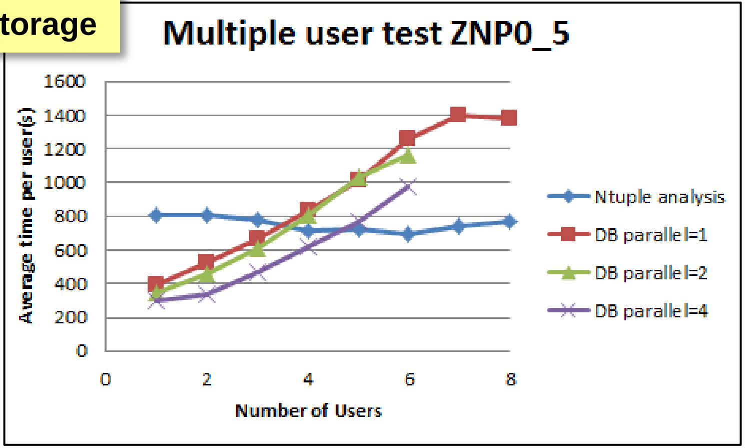
Even with fast local flash disk storage, IO-wait time is still a bottle-neck

Multiple users

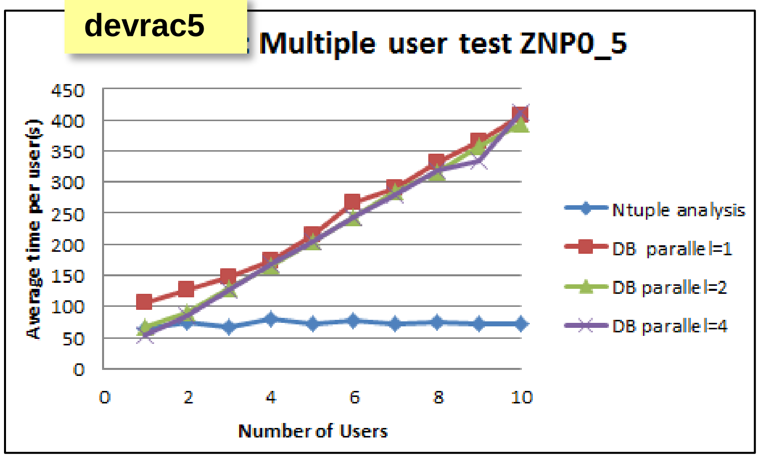
Simulate multiple users accessing the database

Simultaneously run benchmark analysis multiple times with slight variation in cut criteria to create unique queries:

itrac-machines with NFS storage



devrac5



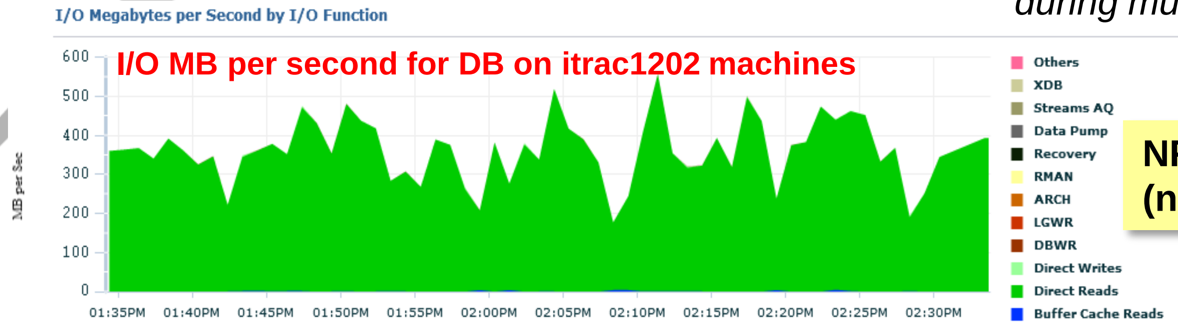
Average analysis time increases rapidly with number of users
Again I/O bottle-neck

Physics Analysis in an Oracle database

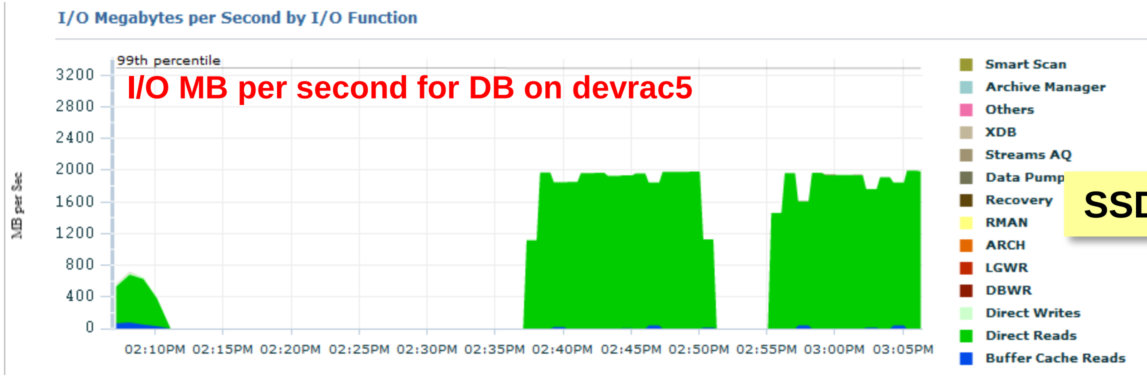
(M. Limper)

IO-wait results from the limit of sequential read on the storage device

performance plots made during multiple user tests



NFS reads up to 500 MB/s (not bad!)



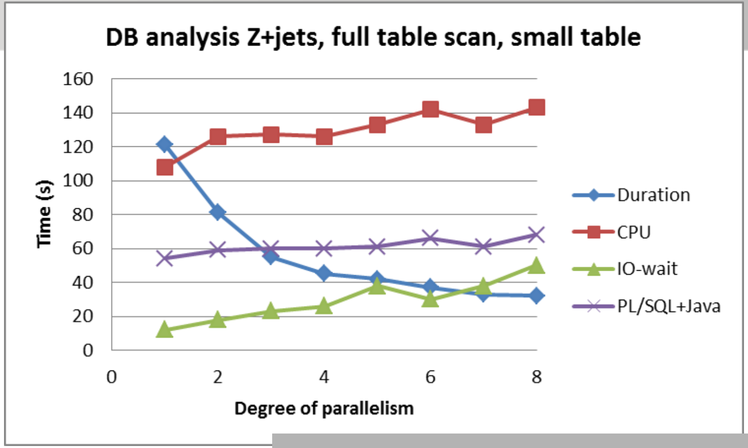
SSD: 2000 MB/s sequential read limit

Test with reduced table content

Small version of the tables: only the variables needed for the benchmark analysis

Z->ll + 2/3/4 jets small				
Table name	columns	k rows	k blocks	size in MB
eventData	3	1662	4.7	37.6
MET_LocHadTopo	5	1662	9.16	73.3
muon	31	1489	29.96	239.7
electron	16	10971	112.64	901.1
jet	12	27931	256.22	2049.8

"jet"-table is 2 GB instead of 45 GB !



Analysis down to 121 seconds
Or 32 seconds with parallel 8 (itrac-setup)

Small table results illustrate the drawback of Oracle DB's row-based storage

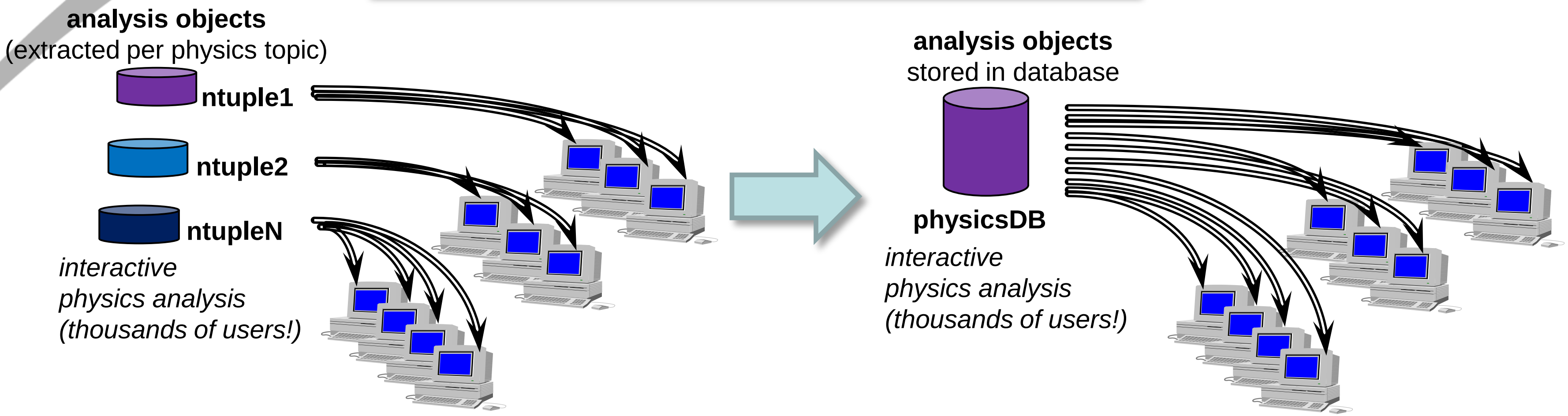
The database is forced to scan through all data in each row to get to the variables needed for analysis

But a real physics analyis database should contain all variables needed for any analysis a user might think of…

Outlook

Physics analysis in an Oracle database ?

Yes it could be done but…



The Oracle database still needs to proof it can handle many users performing their own unique physics analysis studies at the same time

Huge amount of resources needed to build a database used by thousand of physicists and contain all necessary data (simulated, real and multiple production versions)

Conclusion

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 remove the need for separate ntuple production for the different physics groups

Analysis code can be rewritten in SQL

Complicated calculations can be done by external functions

Physics Analysis is I/O intensive, many events are stored but few pass selection

Row-based storage is not ideal when many variables are stored but few variables are needed for a specific analysis, *TTree stored in root was optimized for this!*

It would be interesting to see performance of physics analysis in another type of database (Hadoop?)

Oracle Exadata

Currently preparing to test Physics Analysis on Exadata

Hope to get one week access to an Exadata in february

Oracle Exadata offers interesting features to deal with I/O issues:

- **Smart Scan**
- **Column Projection**
- **Storage Indexes**
- **Hybrid Columnar Compression**