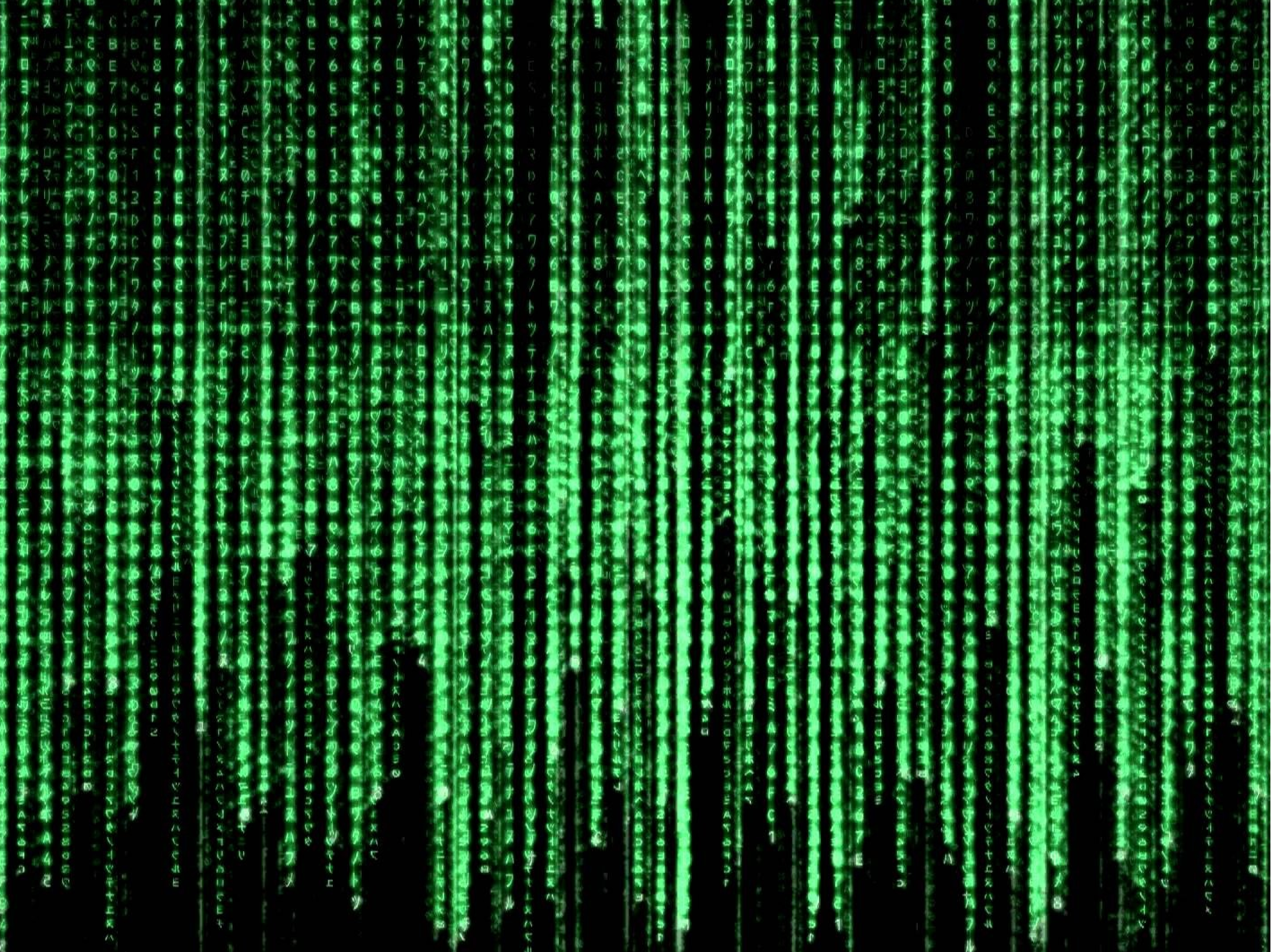
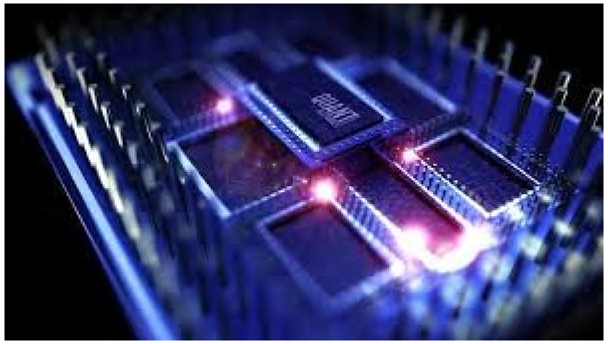


01010111011010000111100100100000011100110110100001101111011101010110110001100100001000000111011101100101001000000110010001101001011100110110001101110101011100110111001100100000011000010110001001101111011101000110011101101000101011110001011010101110100011010010110111001100111000011010000101010



Why should we discuss about computing ?

- Problems have been observed/reported by many of us:
  - Slowness running on UIs (intensive proof usage)
  - Local disks saturated
  - …

  => Need a feedback (survey) to know which problems should be tackle !
- With the restart of the LHC, the volume of data (lumi+PU) to be treated will increase
- Developments have been made by CMS collaboration:
  - miniAOD
  - New CMSSW version (99.3% parallelized)
  - CRAB3
  - …
- Problem of communication btw us & IT group
- **As we are in a period in between publication & restart, it's a window opened to review/change our usage of computing & our tools**

Goals of today's meeting

- Survey of the current **problems** encountered
- Discuss about possible solutions that could be **investigated**
- Define how we could share the efforts
- Define a roadmap for the incoming months
  (@least adapt code to miniAOD)
- Discuss about the interaction with our IT group

- No out-of-the-box solutions
- Overview of the problematics
- Discuss about technical possibilities

Many questions
To be answered !

Wishes box

Simplicity
Reliability
Speed-up the workflow
Data volume reduction
Redundancy (data)
Monitoring tools
Available resources (cpu+disks)
Maintenance/Uniformisation of the UIs

Implications

Code/tools development
Maintenance
Policy
Interactions with IT

Many usage of our computing resources

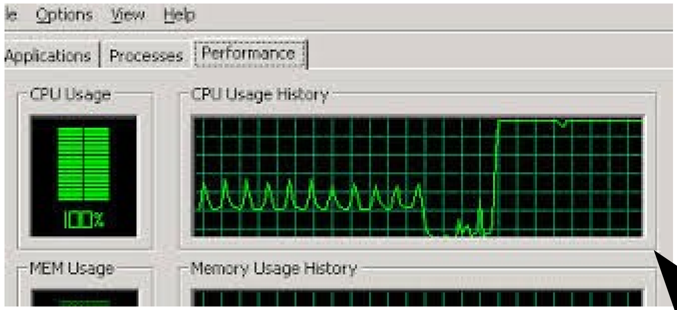
Analysis
(Mainly based on NTuple)

Service work
(b-tagging, trigger, aligment ...)

Limits computation

Event generation

Multi-parametric problems



(Job) management

Software environment

CPU intensive computation

TMVA
C&C optimization

I/O speed

Plots made from
flat root-trees

Madgraph
RooFit/RooStat
MC toys
CLs limits

Running on the whole
MC& CMS data

Large scale computation

Data volume

Job frequency

Importance of
having a prompt output

- Several applications
- Several kind of problems
- Dedicated solutions

Use the “power”
of the **distributed jobs**
(big data & large scale jobs)

Parallelize could be
an option to speed-up
Code execution

Adapt our framework
according to the
dev. of various tools

(re-)**optimization**
of the code

Simplification of
the full analysis chain
(skip some steps)

Better **management**
of our local resources
UIs + storage

Develop tools
to make life easier
for users

Review the **data format**
to speed-up
the reading
(saving space ?)

Change of usage

Modification of codes

Current storage solutions

dpm:

- Large volume
- Can be access through grid, cluster, or ui through xrootd
- dpm-> ui: 40 MB/s (read) + **time for connexion** (stability?)
  - Avoid small files
  - **Tools for management ?**
    merging, recursive cp, listing, size evaluation, bkp management ...

nfs mounted disk:

- Allow sharing of volumes across the Uis
- Performances close to local disk !!
- We use nfs-v3 (nfs-v4 exist and is supposed to be faster ?!)
- Read: ~ 400 MB/s (random)
- Write: ~ 6MB/s

local disk:

- Cannot be shared
- Could be use as scratch
- Supposed to be faster: Read: ~450 MB/s (random)
- Write: ~6/7 MB/s

SSD disk:

- More expensive
- Faster
- Not more than 550 MB/s for random access

Computing resources

CE: ~ 1.5 k nodes

SE: >500 TB

TIER II

11 Uis (7 accessible for CMS)
124 nodes (74)
16 modes max/UI

> 33 TB

UI

TIER II is more suitable for some kind of jobs:

- Big data jobs
- Large scale computation

Could we benefit more often & user-friendly The TIER II ?

- Running on the **grid via crab:**
  - Need data to be published
  - Example: Lighter Minitree (= Ntuple) & running Ntuple analysis on the grid:
    Skimming, cut-flow, babytuple production, ...
- Running with the use of **Proof-on-Demand:**
  - Data stored on dpm
  - Difficulties: loading the librairies
  - Could be helpful to extend current intensive usage of Uis
  - Extend/maintain Kirill's effort
- **WMS jobs** (cf Kirill):
  - Need job monitoring
- **Running on a batch system: pbs**
  - **pbs** is a system recognized by many software to use parallelization
    MadEvent/MadGraph, Theta, ...

Which solution(s) should we try/use/maintain ?
Could we define benchmarks to help user to decide which solution to use ?
Do we have a list of questions to the IT group ?
Do we have requirements ? (pbs, disk accessible via the worker nodes & Uis,...)
Monitoring tools would be helpful !!

« Status » of the UIs

	#nodes	Disk (GB)	Use(%)	
ui1	8	2x900	10%	
ui2	8	2x900	75%	
ui3	4	2x400+900	65%	
ui4	16	900	99%	
ui5	16	900	95%	
ui6	16	800	98%	
ui7 (gd-est)	16	2x900	80%	
ui8	8	2x900	90%	
ui9(gd-est)	8	2x400	55%	
ui10(gd-est)	16	800	6%	
ui11(alice)	16	90	1%	

"Large" resources:

CPU:

76 nodes (124 all included)

Storage:

- "Local disk": 10 TB
- Safe 1: 4.5 TB (99%)
- data2@ui5: 8.5 TB (99%)
- data1@sbgse24: 8.5 TB (38%)

Our usage

CPU:

In average under used (even in peak ?)

Storage:

- "we are using" ~ 23 TB !!
  equivalent to ~ 4E9 Ntuple events …
- Safe1 full ! (code saving …)
- Many of disk ~ full !

Toward a "more efficient" usage of UI ?!

UI sharing ?

7 UIs:

- We could dedicate some of them to a dedicated activity
- The jobs could as much as possible use the local disk (0.8 to 1.8 TB)
  - Faster access
  - Avoid to saturate safe1/sbgse24 access

Use free UI ?

Connexion to a generic UI:

Having a script which allow to connect to a machine depending on the current usage using an alias (as for lxplus)

Do we need tools ?

CPU/RAM monitoring
Alias sbgui

Do we need to keep all those data on those disks ?

- Deletion:
- Avoid many-duplication
  - If needed, dedicate a data volume
- Moving files (from disk to disk)
- Use archive on dpm

Use priority policy (via script)

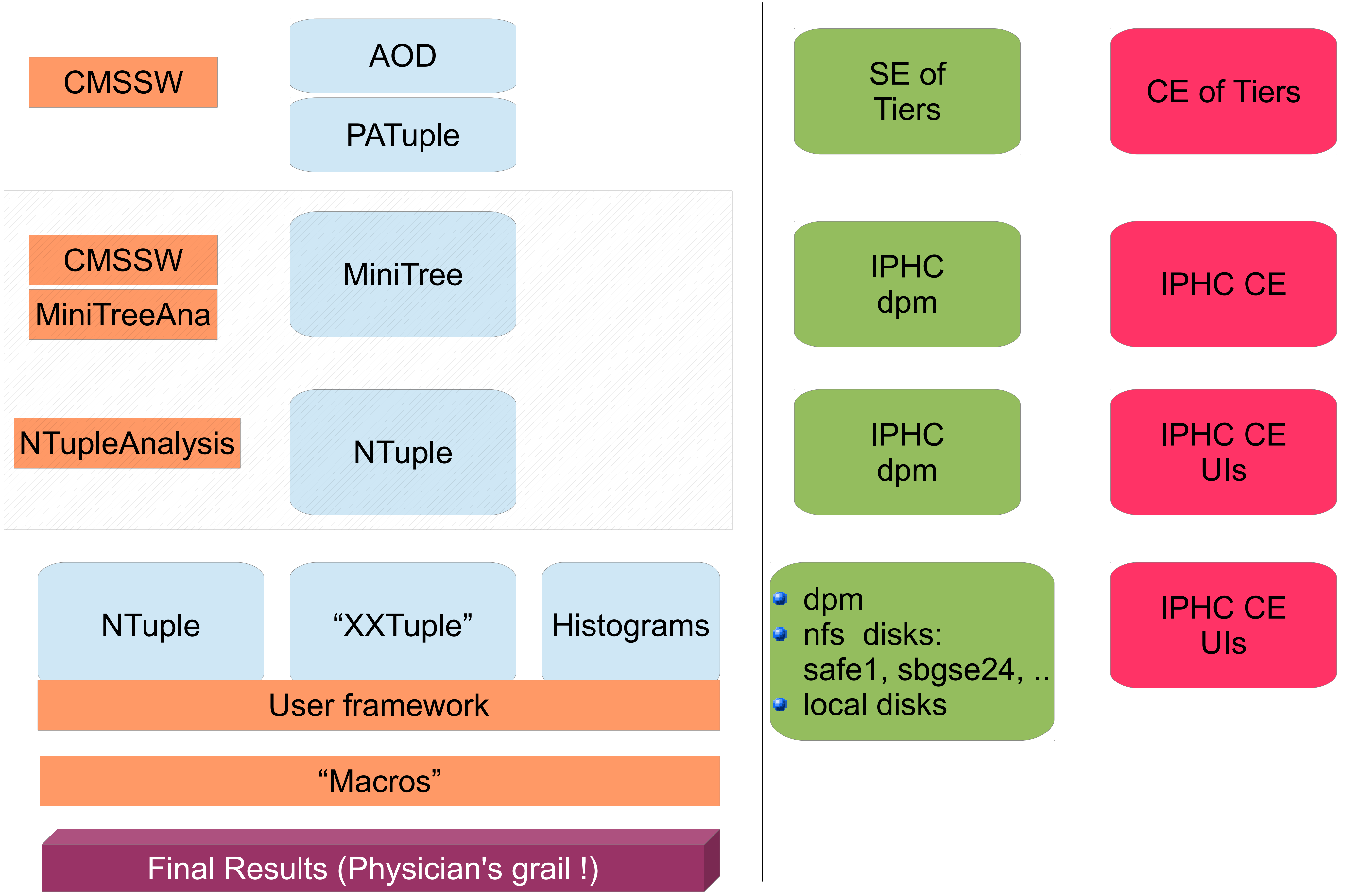
- nice
- ionice

Could we develop tools to help ?

Monitor disk usage
I/O access monitoring
Help deletion (check unread files)
Help archive (tarball ? - recursive cp, …)

NB: exisiting tools – adapt them ?!

Data format | Storage | Computing resources



Comments based on the survey

- **IF** some jobs are performed only once a month &
  IF the time needed is not crucial …
  THEN it might be better to run over dpm files rather than spending time through rfcp …
  - Avoid to overload the local disks
  - Gain using local disk is valuable only if the jobs are performed often
  - Use our cluster (Tier 2/3, via crab/pod/wms …) to increase the #nodes (→ speed-up)

- **IF** some jobs are running locally but take hours &
  IF a prompt feedback would be needed
  THEN you could speed-up things using the // (Proof, scripts with job splitting (/files), std::thread,...)

- **IF** some local jobs using proof are limited to the use of 8-10 nodes *
  IF more nodes would be helpful
  THEN there are some direction to investigate
  - Use local disk to avoid “job concurrency” (try to improve the linearity of the speed-up)
  - Split the dataset into 2 disks and run 2 proof executable (deal about merging @ the end)

  Remark: possibility to use POD-ssh: use several UI at once

Reading tree : bottleneck in analysis jobs

Problem:
A large fraction of most of our analysis jobs is spend in the line:

tree->GetEntry(i)

It can represent btw 10 to 90% of the job's time !

What is done "behind" that line ?

- I/O access
- Management of the data by ROOT (CPU usage)

How can we improve ??

I/O access:

- use local disk
- reduce data size (/evt)
- Use a skimming (branches/evts)

ROOT management:

- Do not load useless branches
- Read on demand
- Buffer size (config)
- Compression (config)
- Splitting (config)
- Data format (faster if simple) ***DEV***
- Unzipping (could be parallelized)

MiniAOD: 30-50 kB/evt
Max 600 kevts/min = 36 Mevts/hour
(If stored on a given HD & if not computation …)

Ntuple: 5-10 kB/evt
Could still gain x5 (or more)

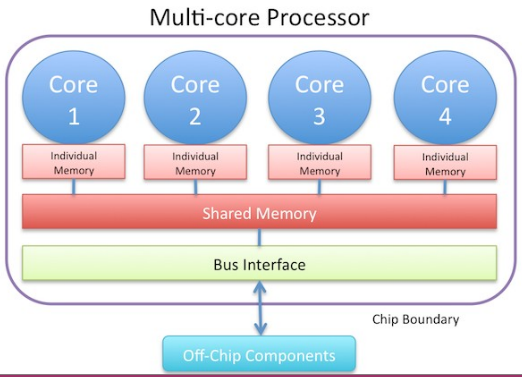
Babytuple: 0.2 kB/evt
Analyze@max O100Kevts/sec

Do we review our data format ?
Do we invest time to optimize tree reading ?

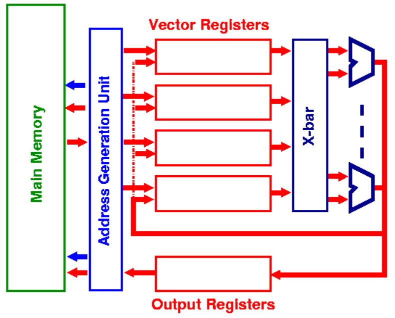
Welcome to the real of parallel computing

Cluster: many machines

Multi-threading



Vector processor



Operation in // (cache line)
X double (X=2 to 8)
2X float or int
4X short
8X bool

I – Many machines

- Splitting of jobs per **file**:
  - Ex: wms jobs
  - Might require job managements via scripts
- Splitting of jobs per **event**:
  - Ex:PROOF (Pod - Pod-ssh)

II- Multi-threading

- Tools: std:thread (c++11) or open-mp
- Applications:
  - Parallelization of algorithms:
    Jets, electrons, muons … selections done in //
  - Operation on objects of a collections done in //:
    Ex: Applying correction to jets

III Vectorialization:

- Computation of functions
  Ex: sqrt, cos, ...
- Treating vector of bool/int/float/double

“the roadmap”

- **Survey of the current usage & the problems that need to be “solved”**
  - Define list of priorities

- **Define how we share the efforts**
  - Common data-format
  - Common analysis framework (or block of software tools)
  - Common scripts

Thibaut can help us in part of those tasks !

- **Usage of UI:**
  - Do we agree on a new way to use the resources ? (cpu+disk)

- **Increase the usage of TIER II/III:**
  - Which solutions to tests ? (PoD, pbs, ...)
  - Which tools are needed ?

- Define how we want to interact with IT group in the future
- Define a list of request we have for the IT group

“the roadmap”

On the analysis side

- **MiniAOD:**
  Who will follow MiniAOD dev ?
  Who will adap the framework ?
  Do we still need MiniTrees ?

- **Which scheme to adapt ? (MiniAOD,MiniTree, Ntuple, …)**

- **Do we need to revisit the data format ?**
  Need time, development, modification of current macros (? not necessarily)
  We could hopefully same a lot of time @ analysis level

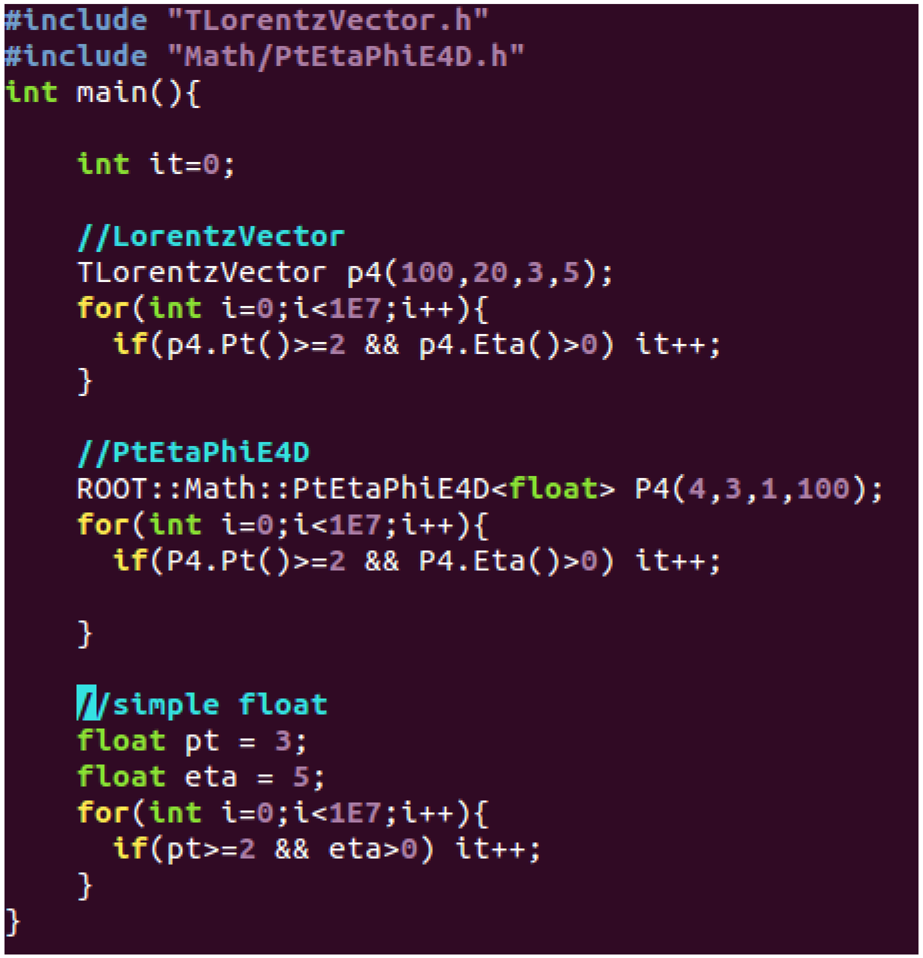
- **Do we want to share analysis tools ?**
  How to maintain them ? (documentation ?)
  Do we need a policy for code-development

- **Do we want to improve code efficiency ?**
  Need code profiling
  Requirement: keeping tools user-friendly
  Providing guidelines to all code-developers

Ex: TLorentzVector is not efficient
We're using Pt() & Eta() which need computations:
sqrt(Perp2)
0.5*log((m+fZ)/(m-fZ))

Developments would/will take time

… and maintenance looks like running a marathon ..



```
#include "TLorentzVector.h"
#include "Math/PtEtaPhiE4D.h"
int main(){

    int it=0;

    //LorentzVector
    TLorentzVector p4(100,20,3,5);
    for(int i=0;i<1E7;i++){
      if(p4.Pt()>=2 && p4.Eta()>0) it++;
    }

    //PtEtaPhiE4D
    ROOT::Math::PtEtaPhiE4D<float> P4(4,3,1,100);
    for(int i=0;i<1E7;i++){
      if(P4.Pt()>=2 && P4.Eta()>0) it++;

    }

    //simple float
    float pt = 3;
    float eta = 5;
    for(int i=0;i<1E7;i++){
      if(pt>=2 && eta>0) it++;
    }
}
```

10 times faster

20 times faster

User name	Task	Data format	Data storage	Data size	CPU-resources	Nb nodes	Frequency	Estimated time	Observed problems	Wishes/comments
aaubin	1lepton Stop Analysis (MiniTree prod)	SBG MiniTree	DPM	O(15 To) (~40Ko/evts ?)	Grid	1 node / 20k evts	A few times on the whole analysis	2-8 weeks (depending of grid and user bugs)	- Random crashes, need a lot of monitoring	- A tool to check disk usage on dpm easily would be great (I tried to implement a prototype but it's not complete)
aaubin	1lepton Stop Analysis (NTuple prod)	SBG Ntuple	DPM + data4 & 1	~2.5 To	Grid (sbg only ?)	1 node / 500k evts	A few times on the whole analysis	24 - 48 hours		
aaubin	1lepton Stop Analysis (babyTuple prod)	1leptonStop babyTuples	data4	~75 Go	UI / Proof	O(10)	0.5 - 2 per months	24 - 48 hours	- Need to copy nTuples to UI to run PROOF - Sometimes no CPU available on UI - Sometimes no disk space available on data4	- A way to avoid copying nTuples to the UI - Lighter nTuples for this step to be faster :)
blochd	b-tagging	BTagAnalyzer Ntuple	DPM and UIs	~2 To (for all 7-8 TeV)	UI	usually 1 node	several per week on a few Go only	1-8 hours / week	usual crab pbs to create the ntuples, then ok	- a tool to get the disk usage of each user on dpm ...
mbuttign	Monotop AOD Production	AODSIM	DPM	~750 Go (~250Ko/evt)	Grid	1 node	Ideally only one	Weeks (depending on grid bugs)	Very random crashes	
mbuttign	Monotop NTuple Production	SBG NTuple	DPM + UIs	~1.5 Go (0.5 Ko/evt ?)	Grid or UIs	1 node	Ideally only one	Few hours		
caroline	b-tagging	BTagAnalyzer Ntuple	DPM & UIs	heavier than Daniel (more	prod on Grid, lec	1 node	several per week on a short period	few hours		a better way to use the UI between us, 1 UI/ person? how to ir
mbuttign	Monotop Analysis	BabyTuples	UIs	< 5 Go	UI / Proof	Usually 8 nodes	1 /week	Less than 24 hours		
caroline	1lepton stop analysis	1leptonStop babyTuples	using Alex ntuples		Ui		access several times per week		the previous versions of the babytuples are not	have a history twiki page to remember the change in the differ
ttH team	ttH (not extensively tested so far)	SBG minitrees	DPM		Grid		From time to time			
ttH team	ttH (not extensively tested so far)	SBG ntuples	DPM + disks		Grid		From time to time			avoid copying ntuples from /dpm
Anne-Cath	HLT b-tagging	HLT output	DPM		Grid /UI	1 node on ui	several times a week/ when neede	few hours	path to dpm changed recently/ problems with r	better maintenance of uis (xterm, evince...)
ttH team	ttH analysis (not extensively tested so f	analysis histos	disk		ui/proof	O(10)			proof painful to debug ! Can't submit everything	would like to avoid using proof / batch system instead !
tth team	matrix element method	text files	disk		several nodes			CPU extensive	need PBS batch system !	
kskovpen	b-tagging,tZq,ttH,etc.	BTagAnalyzer Ntuple, SBG Ntuple	DPM	Not running into issues	Tier2	All I can find	Daily & Nightly	1-2 hours	Tier3-related, hence many random issues	have to use Tier2 storage as this is the only way to get use of
										would be really-really-really nice to have Tier3