

7. 4. 6. 8. 2. 1. 3. 5. 7. 9. 11. 13. 15. 17. 19. 21. 23. 25. 27. 29. 31. 33. 35. 37. 39. 41. 43. 45. 47. 49. 51. 53. 55. 57. 59. 61. 63. 65. 67. 69. 71. 73. 75. 77. 79. 81. 83. 85. 87. 89. 91. 93. 95. 97. 99. 101. 103. 105. 107. 109. 111. 113. 115. 117. 119. 121. 123. 125. 127. 129. 131. 133. 135. 137. 139. 141. 143. 145. 147. 149. 151. 153. 155. 157. 159. 161. 163. 165. 167. 169. 171. 173. 175. 177. 179. 181. 183. 185. 187. 189. 191. 193. 195. 197. 199. 201. 203. 205. 207. 209. 211. 213. 215. 217. 219. 221. 223. 225. 227. 229. 231. 233. 235. 237. 239. 241. 243. 245. 247. 249. 251. 253. 255. 257. 259. 261. 263. 265. 267. 269. 271. 273. 275. 277. 279. 281. 283. 285. 287. 289. 291. 293. 295. 297. 299. 301. 303. 305. 307. 309. 311. 313. 315. 317. 319. 321. 323. 325. 327. 329. 331. 333. 335. 337. 339. 341. 343. 345. 347. 349. 351. 353. 355. 357. 359. 361. 363. 365. 367. 369. 371. 373. 375. 377. 379. 381. 383. 385. 387. 389. 391. 393. 395. 397. 399. 401. 403. 405. 407. 409. 411. 413. 415. 417. 419. 421. 423. 425. 427. 429. 431. 433. 435. 437. 439. 441. 443. 445. 447. 449. 451. 453. 455. 457. 459. 461. 463. 465. 467. 469. 471. 473. 475. 477. 479. 481. 483. 485. 487. 489. 491. 493. 495. 497. 499. 501. 503. 505. 507. 509. 511. 513. 515. 517. 519. 521. 523. 525. 527. 529. 531. 533. 535. 537. 539. 541. 543. 545. 547. 549. 551. 553. 555. 557. 559. 561. 563. 565. 567. 569. 571. 573. 575. 577. 579. 581. 583. 585. 587. 589. 591. 593. 595. 597. 599. 601. 603. 605. 607. 609. 611. 613. 615. 617. 619. 621. 623. 625. 627. 629. 631. 633. 635. 637. 639. 641. 643. 645. 647. 649. 651. 653. 655. 657. 659. 661. 663. 665. 667. 669. 671. 673. 675. 677. 679. 681. 683. 685. 687. 689. 691. 693. 695. 697. 699. 701. 703. 705. 707. 709. 711. 713. 715. 717. 719. 721. 723. 725. 727. 729. 731. 733. 735. 737. 739. 741. 743. 745. 747. 749. 751. 753. 755. 757. 759. 761. 763. 765. 767. 769. 771. 773. 775. 777. 779. 781. 783. 785. 787. 789. 791. 793. 795. 797. 799. 801. 803. 805. 807. 809. 811. 813. 815. 817. 819. 821. 823. 825. 827. 829. 831. 833. 835. 837. 839. 841. 843. 845. 847. 849. 851. 853. 855. 857. 859. 861. 863. 865. 867. 869. 871. 873. 875. 877. 879. 881. 883. 885. 887. 889. 891. 893. 895. 897. 899. 901. 903. 905. 907. 909. 911. 913. 915. 917. 919. 921. 923. 925. 927. 929. 931. 933. 935. 937. 939. 941. 943. 945. 947. 949. 951. 953. 955. 957. 959. 961. 963. 965. 967. 969. 971. 973. 975. 977. 979. 981. 983. 985. 987. 989. 991. 993. 995. 997. 999. 1001. 1003. 1005. 1007. 1009. 1011. 1013. 1015. 1017. 1019. 1021. 1023. 1025. 1027. 1029. 1031. 1033. 1035. 1037. 1039. 1041. 1043. 1045. 1047. 1049. 1051. 1053. 1055. 1057. 1059. 1061. 1063. 1065. 1067. 1069. 1071. 1073. 1075. 1077. 1079. 1081. 1083. 1085. 1087. 1089. 1091. 1093. 1095. 1097. 1099. 1101. 1103. 1105. 1107. 1109. 1111. 1113. 1115. 1117. 1119. 1121. 1123. 1125. 1127. 1129. 1131. 1133. 1135. 1137. 1139. 1141. 1143. 1145. 1147. 1149. 1151. 1153. 1155. 1157. 1159. 1161. 1163. 1165. 1167. 1169. 1171. 1173. 1175. 1177. 1179. 1181. 1183. 1185. 1187. 1189. 1191. 1193. 1195. 1197. 1199. 1201. 1203. 1205. 1207. 1209. 1211. 1213. 1215. 1217. 1219. 1221. 1223. 1225. 1227. 1229. 1231. 1233. 1235. 1237. 1239. 1241. 1243. 1245. 1247. 1249. 1251. 1253. 1255. 1257. 1259. 1261. 1263. 1265. 1267. 1269. 1271. 1273. 1275. 1277. 1279. 1281. 1283. 1285. 1287. 1289. 1291. 1293. 1295. 1297. 1299. 1301. 1303. 1305. 1307. 1309. 1311. 1313. 1315. 1317. 1319. 1321. 1323. 1325. 1327. 1329. 1331. 1333. 1335. 1337. 1339. 1341. 1343. 1345. 1347. 1349. 1351. 1353. 1355. 1357. 1359. 1361. 1363. 1365. 1367. 1369. 1371. 1373. 1375. 1377. 1379. 1381. 1383. 1385. 1387. 1389. 1391. 1393. 1395. 1397. 1399. 1401. 1403. 1405. 1407. 1409. 1411. 1413. 1415. 1417. 1419. 1421. 1423. 1425. 1427. 1429. 1431. 1433. 1435. 1437. 1439. 1441. 1443. 1445. 1447. 1449. 1451. 1453. 1455. 1457. 1459. 1461. 1463. 1465. 1467. 1469. 1471. 1473. 1475. 1477. 1479. 1481. 1483. 1485. 1487. 1489. 1491. 1493. 1495. 1497. 1499. 1501. 1503. 1505. 1507. 1509. 1511. 1513. 1515. 1517. 1519. 1521. 1523. 1525. 1527. 1529. 1531. 1533. 1535. 1537. 1539. 1541. 1543.



0101011101101000011110010010000001110011011  
0100001101111011101010110110001100100001000  
0001110111011001010010000001100100011010010  
1110011011000110111010101110011011100110010  
0000011000010110001001101111011101010111010  
0001000000110001101101111011011010111000001  
1101010111010001101001011011100110011100001  
10100001010



# 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**

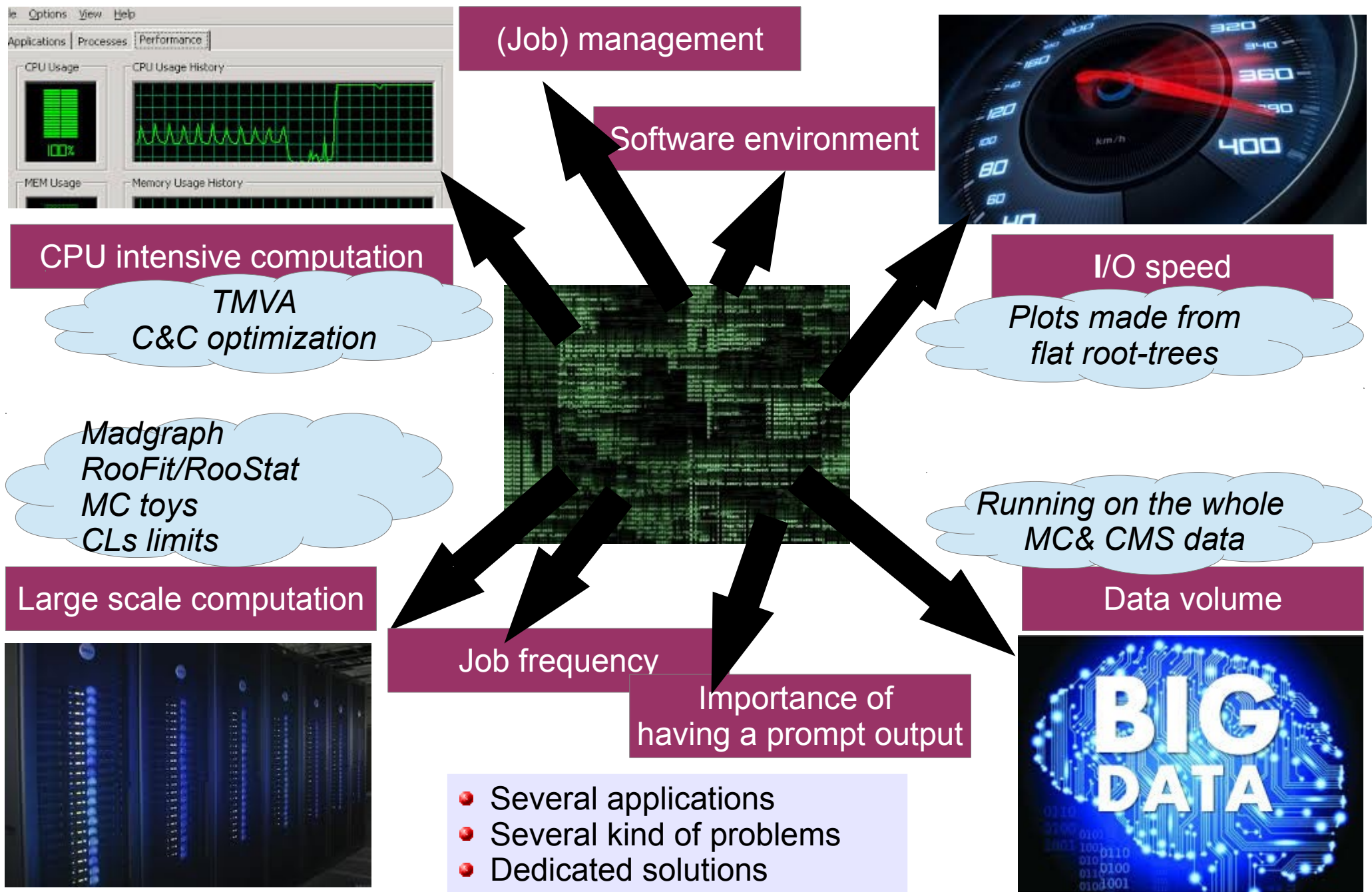
(b-tagging, trigger, alignment ...)

## **Limits computation**

## **Event generation**



# Multi-parametric problems





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  
Uls + 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

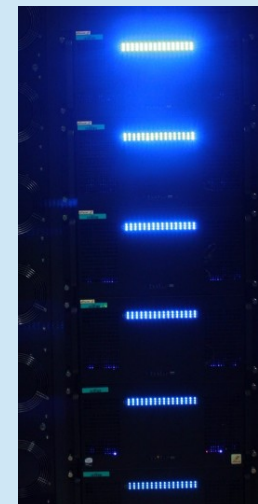


# Computing resources

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 lxxplus)

## 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

## Could we develop tools to help ?

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

NB: existing tools – adapt them ?!

## Use priority policy (via script)

- nice
- ionice

# Data format

CMSSW

AOD

PATuple

CMSSW

MiniTreeAna

MiniTree

NTupleAnalysis

NTuple

NTuple

“XXTuple”

Histograms

User framework

“Macros”

Final Results (Physician's grail !)

# Storage

SE of  
Tiers

IPHC  
dpm

IPHC  
dpm

- dpm
- nfs disks:  
safe1, sbgse24, ..
- local disks

# Computing resources

CE of Tiers

IPHC CE

IPHC CE  
ULs

IPHC CE  
ULs

# 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 rfcv ...
  - ➔ 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 kevt/s/min = 36 Mevt/s/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



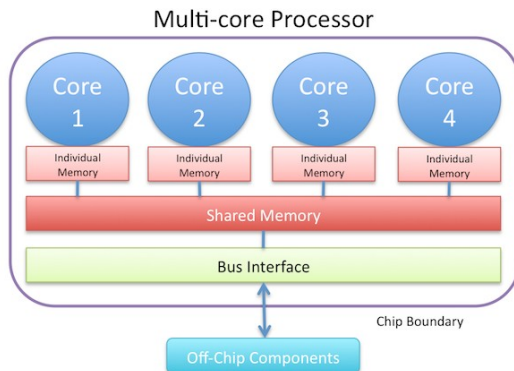
Do we review our data format ?  
Do we invest time to optimize tree  
reading ?

# Welcome to the real of parallel computing

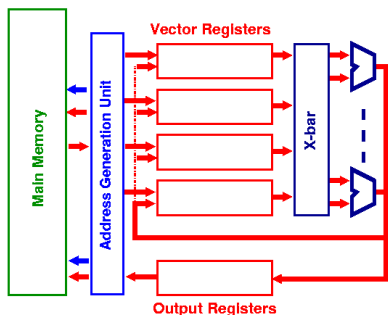
Cluster: many machines



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 adapt 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 save 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:

$\text{sqrt}(\text{Perp}^2)$

$0.5 * \log((m+fZ)/(m-fZ))$

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<br>- Sometimes no CPU available on UI<br>- Sometimes no disk space available on data4 | - A way to avoid copying nTuples to the UI<br>- 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, less on UI) | prod on Grid, less on UI | 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 improve it?  |
| 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 available   | have a history twiki page to remember the change in the different versions   |
| 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 needed  | few hours                                   | path to dpm changed recently/ problems with rucio   | better maintenance of uis (xterm, evince...)   |
| tth team  | tth analysis (not extensively tested so far) | 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 Tier3  |
|           |  |                                 |                    |   |                          |                    |                                    |   |   | would be really-really-really nice to have Tier3   |