Impact des évolutions matérielles récentes sur l'environnement logiciel de la physique HEP

IN2P3, Aix les Bains
16 Juin 2011

René Brun, CERN

plan

- Pour cette présentation j'ai sélectionné 2 sujets très différents qui peuvent avoir un impact sur un centre de calcul dans le court ou moyen terme
  - Accès aux données a travers un WAN combiné avec un cache sur un LAN.
  - Exemple de projet exploitant le parallélisme dans un programme de simulation.

Accès aux données a travers un WAN

Modèle classique sur la grille

- Data: T0 → T1 –> T2
- Les jobs sont envoyés ou sont les données.
- Les statistiques montrent que dans le cas de la reconstruction un fichier est utilisé 0.98 fois!
- Ce modèle implique une lecture sur T1, une écriture sur T2 et une lecture sur T2.
- Ce modèle était impératif car nous n'avions pas résolu l'accès efficace aux données a travers un WAN.

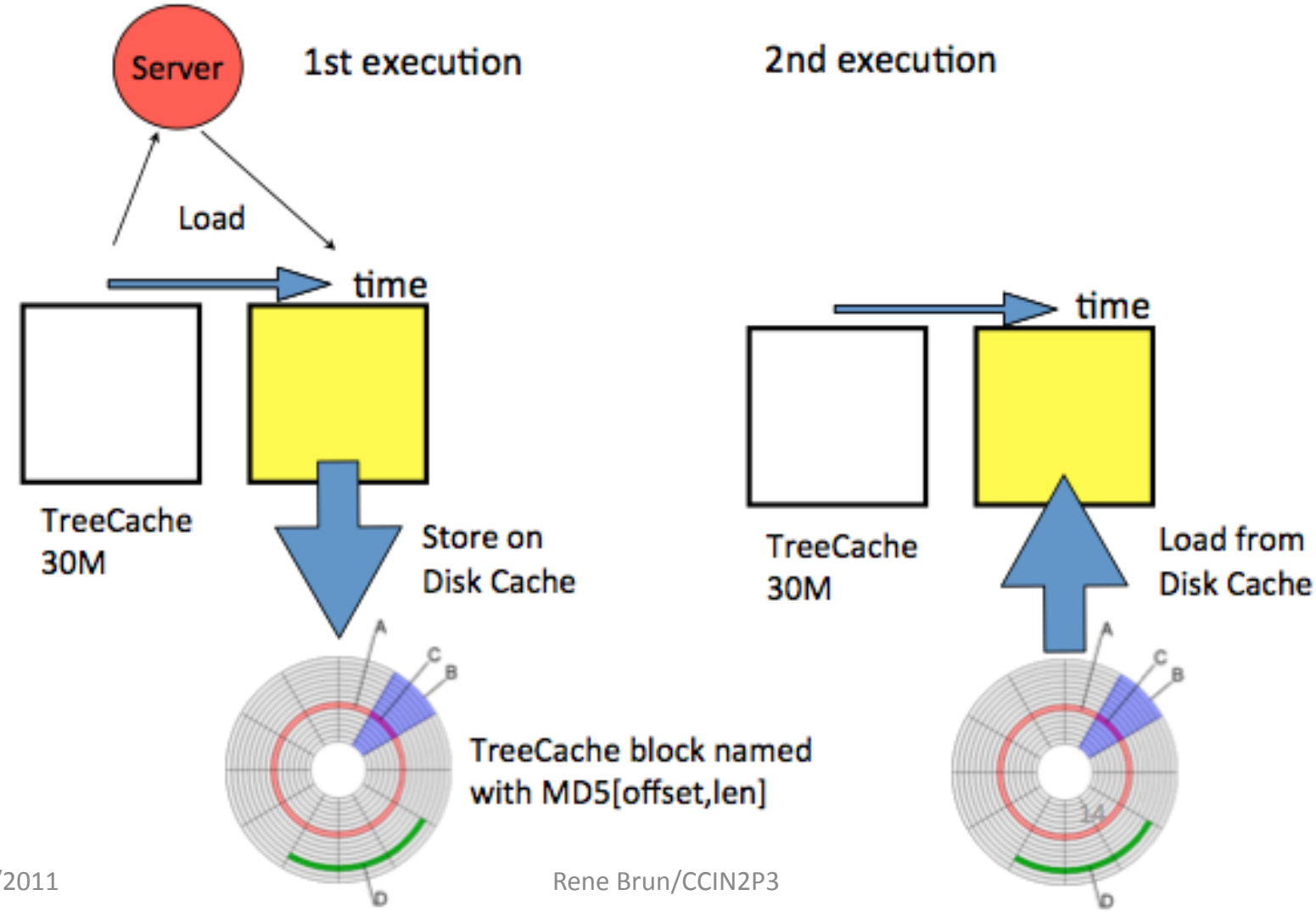
Nouveau modèle

- Les données restent sur T1.
- Les jobs sur T2 accèdent directement le T1 en transférant des gros blocks (30 MB) qui sont ensuite cachés (optionnellement sur le LAN du T2).
- Performances actuelles
  - 6 a 7 MB/s sur un WAN avec latence de 180ms
  - 8 a 12 MB/s sur un WAN avec latence de 10 ms

Nouveau modèle (2)

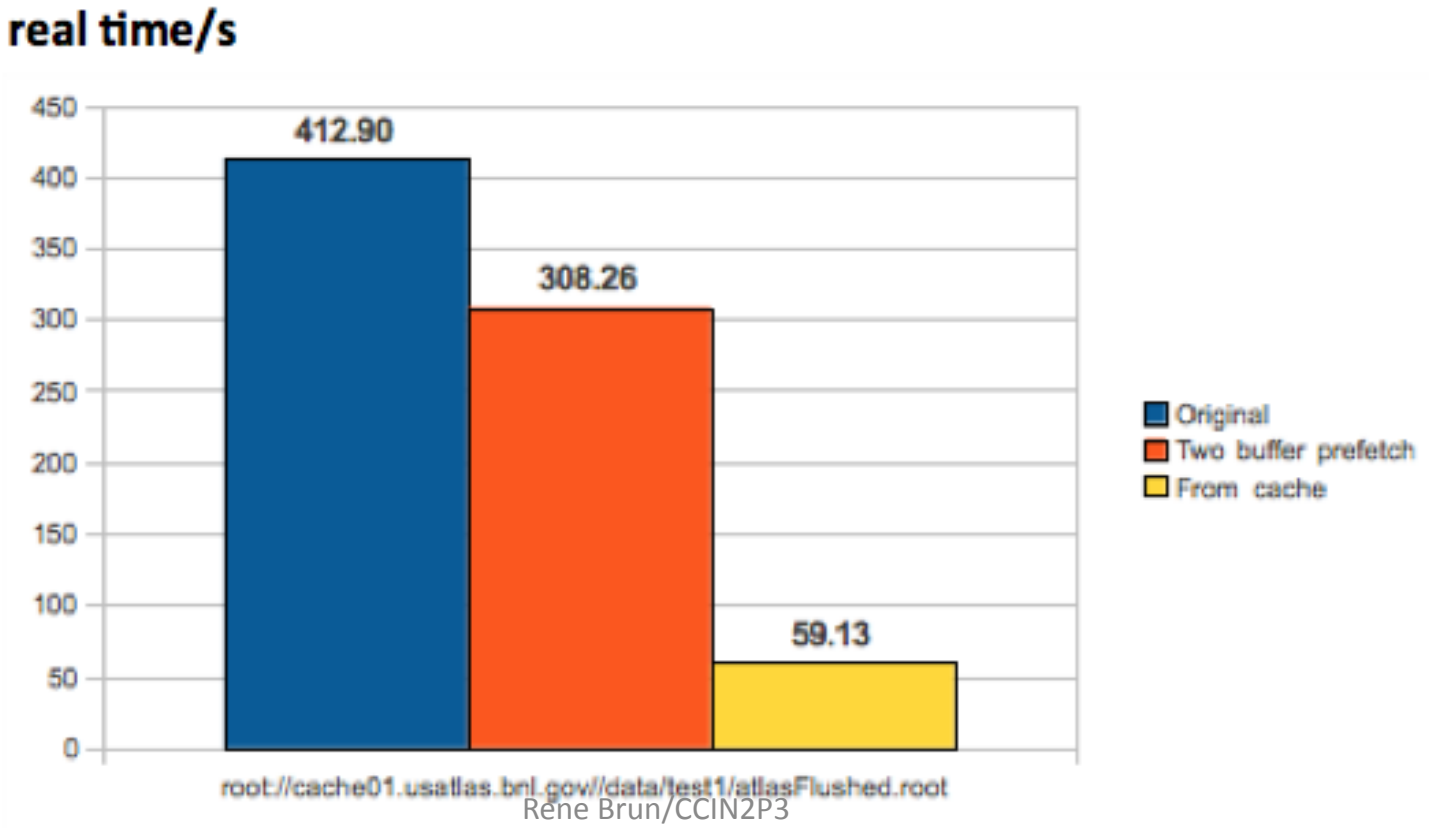
- Le cache local (30MB) correspond a un TreeCache de ROOT.
- Ce cache est sauvé comme fichier local avec xrootd (mais pourrait aussi l'être avec Lustre, NFS4, etc).
- Le cache mémoire de xrootd est important pour la performance ainsi que le TreeCache read-ahead exploitant le vector-read du système.
- Ce nouveau modèle pourrait grandement être optimisé en performance en adaptant dynamiquement les paramètres tcp/ip.

Asynchronous pre-fetching of buffers



Nouveau modèle (3)

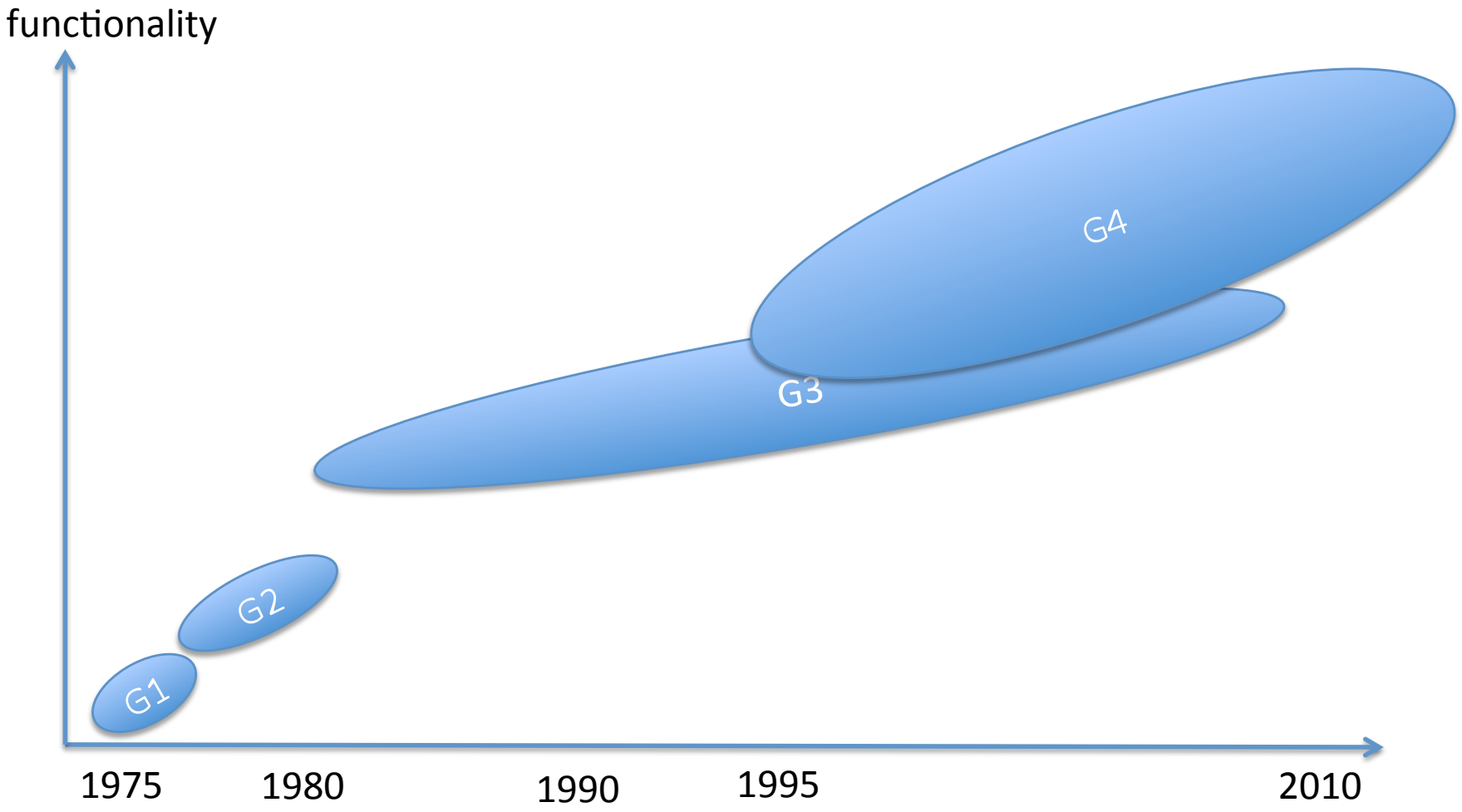
- Actuellement en test et pré-production avec ATLAS.
- Contactez Dirk Duellmann, Andreas Peters.
- Le même modèle intéresse CMS pour T2←>T3



Nouvelle stratégie pour la simulation exploitant les architectures parallèles

GEANT --→ GEANT5 ?

The GEANT versions (III)



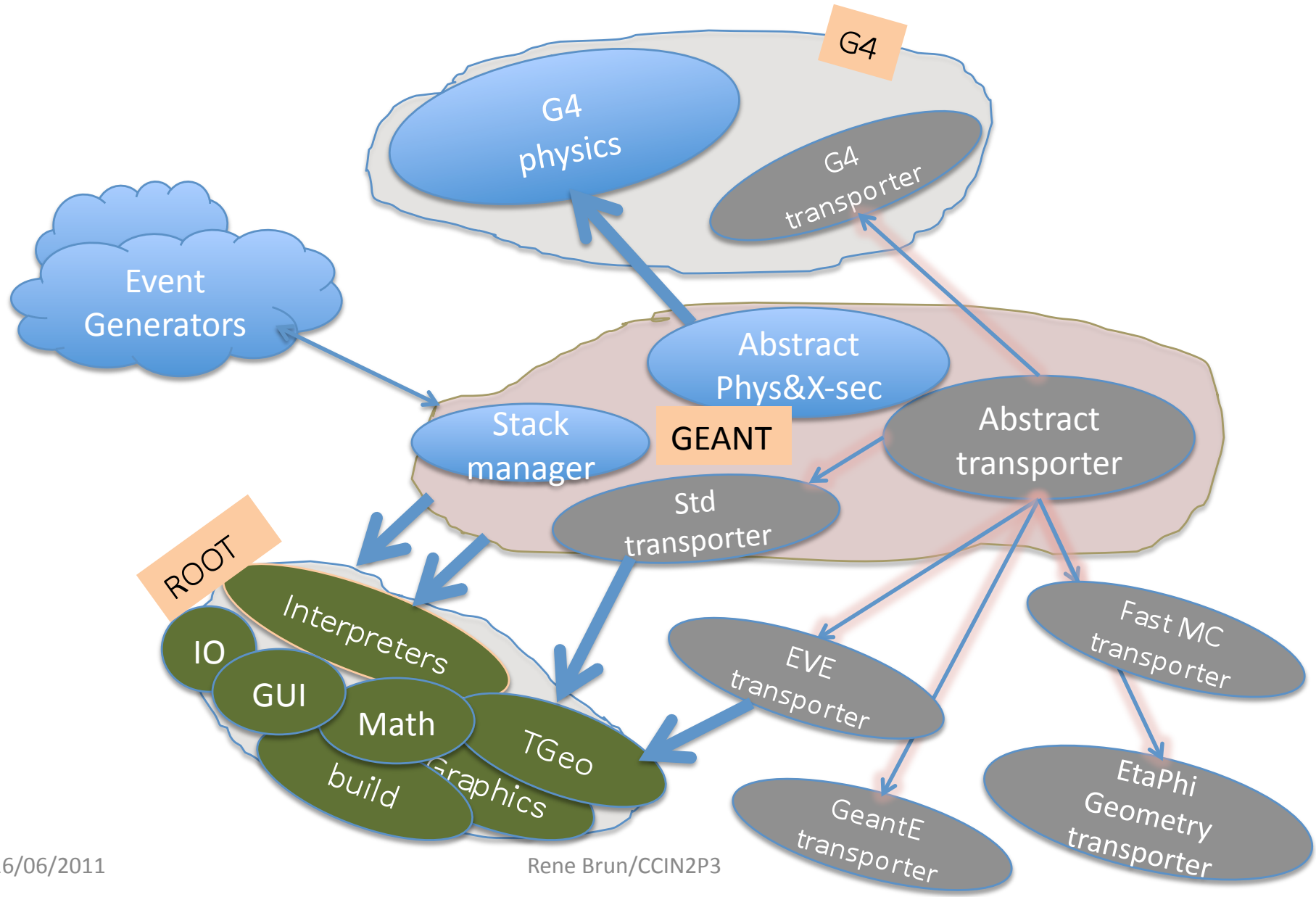
Le projet GEANT→GEANT5

- Ce projet lancé en Décembre 2010 a pour but de dessiner une nouvelle infrastructure de simulation:
  - En combinant les « fast » et « full » Montecarlo dans la même architecture.
  - En proposant de nouvelles structures pour exploiter de façon dynamique les systèmes avec un grand nombre de cœurs ou gpus.

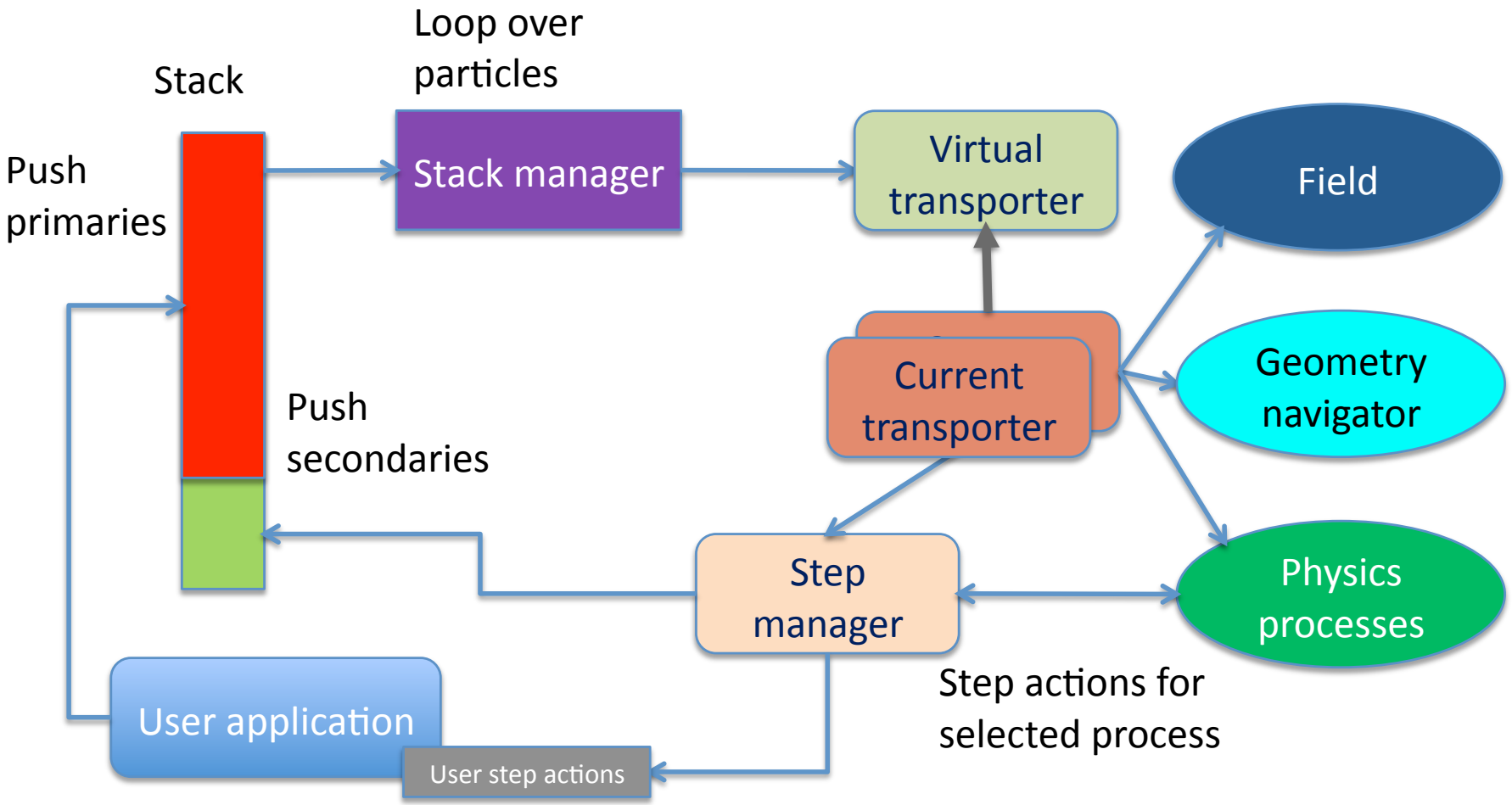
« fast » et « full » ensemble

- Toutes les expériences sont actuellement en train de mettre au point un système de « fast » MC qui soit entre 10 et 100 fois plus rapide que G4 afin d'exécuter dans le même temps que la reconstruction et ainsi minimiser les problèmes de « data management ».
- Le nouveau prototype de GEANT propose un système de classes qui puisse combiner les 2.

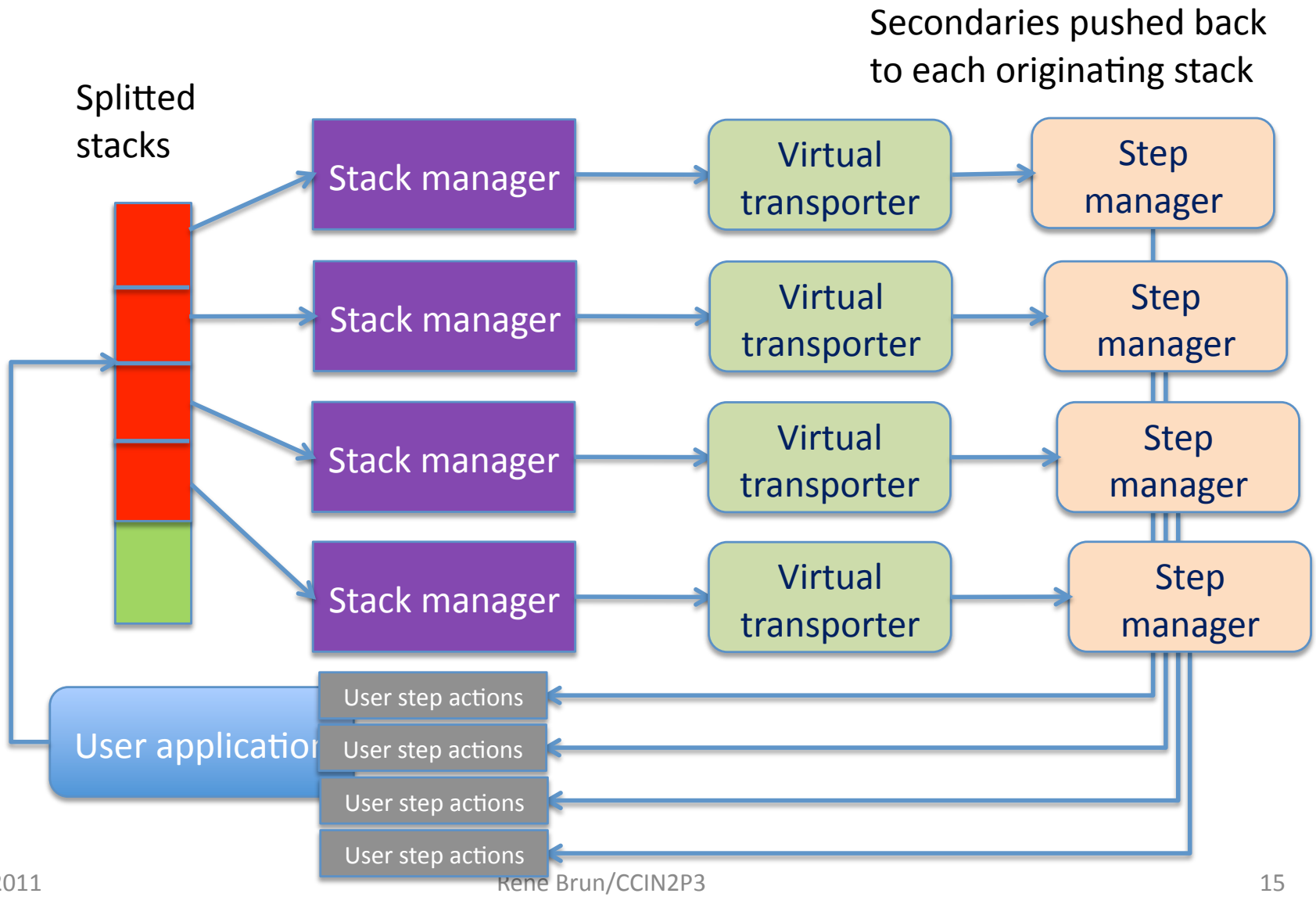
New GEANT in one picture



Event loop and stacking



Parallel approach



Penser parallèle

- Depuis l'origine du computing en HEP nous avons toujours pensé "séquentiel" dans nos programmes.
- Par contre les CC et OS ont toujours pensé "parallèle" (jobs)
- L'avènement du GRID/CLOUD a été un frein a penser parallèle dans nos programmes.
- L'avènement des nouvelles architectures va nous forcer a penser parallèle (question ou affirmation?)

Remarques

- Je ne lis pas dans la boule de Crystal.
- Nous nous sommes très souvent trompés dans le passé dans nos prédictions sur le parallélisme
- Vectorisation sur Cray, cyber205, IBM3081
- Les MPP au début des années 1990.
- Quelquefois nous avons été pris pour des cobayes par les vendeurs.

Parallélisme et Parallélisme

- Nous avons plusieurs niveaux évidents de parallélisme (jobs, évènements, traces) bien adaptés a un nombre restreint de processeurs, mais pas efficaces du point de vue mémoire ou I/O dans le cas d'un très grand nombre de processeurs.
- D'autre part a un niveau plus bas, nous n'avons jamais été en mesure d'exploiter le pipeline d'un processeur (vectorisation).

Job parallélisme sur la grille

- Cela marche bien, mais plusieurs questions ouvertes:
  - « merge » des fichiers résultats dans les jobs d'analyse.
  - Faut-il donner 8 cœurs a un job ou exécuter 8 jobs sur le processeur (sujet a bagarres!)
  - Comment exploiter au mieux la mémoire, les accès disques et réseau sur un système avec >= 8 cœurs?

Evènements en parallèle

- Essentiel pour l'acquisition des données.
- Inutile pour la production ou simulation.
- Peut être utile pour l'analyse. Par exemple PROOF traite des paquets d'évènements en parallèle.
- Ne résous pas les problèmes de mémoire ou I/O

Multi-threading

- Une façon d'exploiter plusieurs cœurs en parallèle (réduction importante de mémoire).
- Mais il est très difficile de transformer un programme conçu pour être séquentiel a cause des accès simultanés aux structures en mémoire et des entrées/sorties.
- Problèmes inextricables avec les mutex et la loi de Amdhal. Un pourcentage infime de « sequentialité » va détruire la performance sur un système avec un grand nombre de processeurs.

La loi de Amdhal

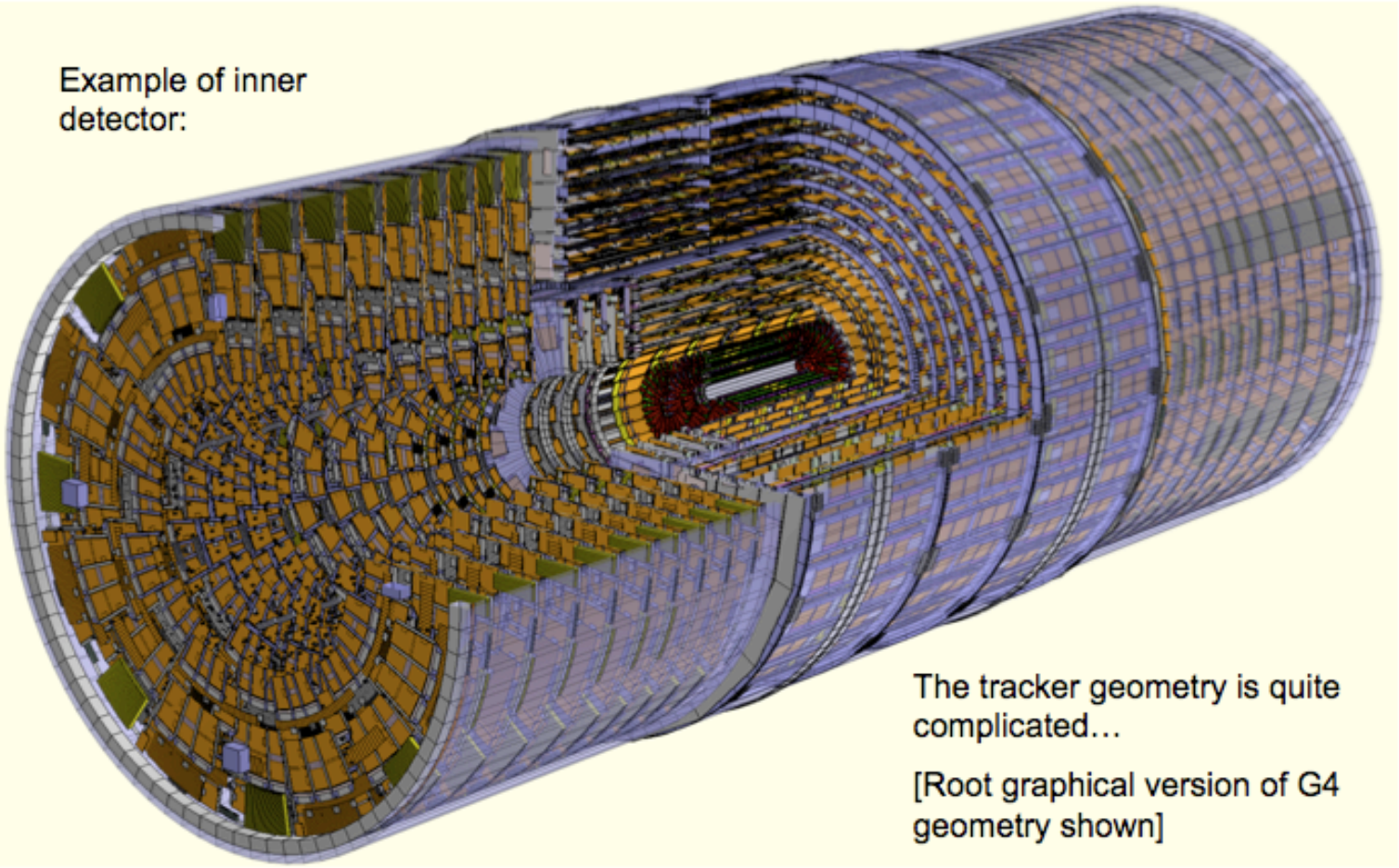
Simulation d'un détecteur (actuellement)

- Comporte 2 phases
  - Le « tracking » particule après particule.
    - Chaque particule traverse les détecteurs et a chaque « step » le step manager est appellé ou l'utilisateur enregistre les hits et décide de continuer ou non avec la particule et ses filles.
  - Ensuite les « hits » sont convertis en « digits » en tenant compte de toutes les particules et de l'électronique des détecteurs.

Geometries complexes

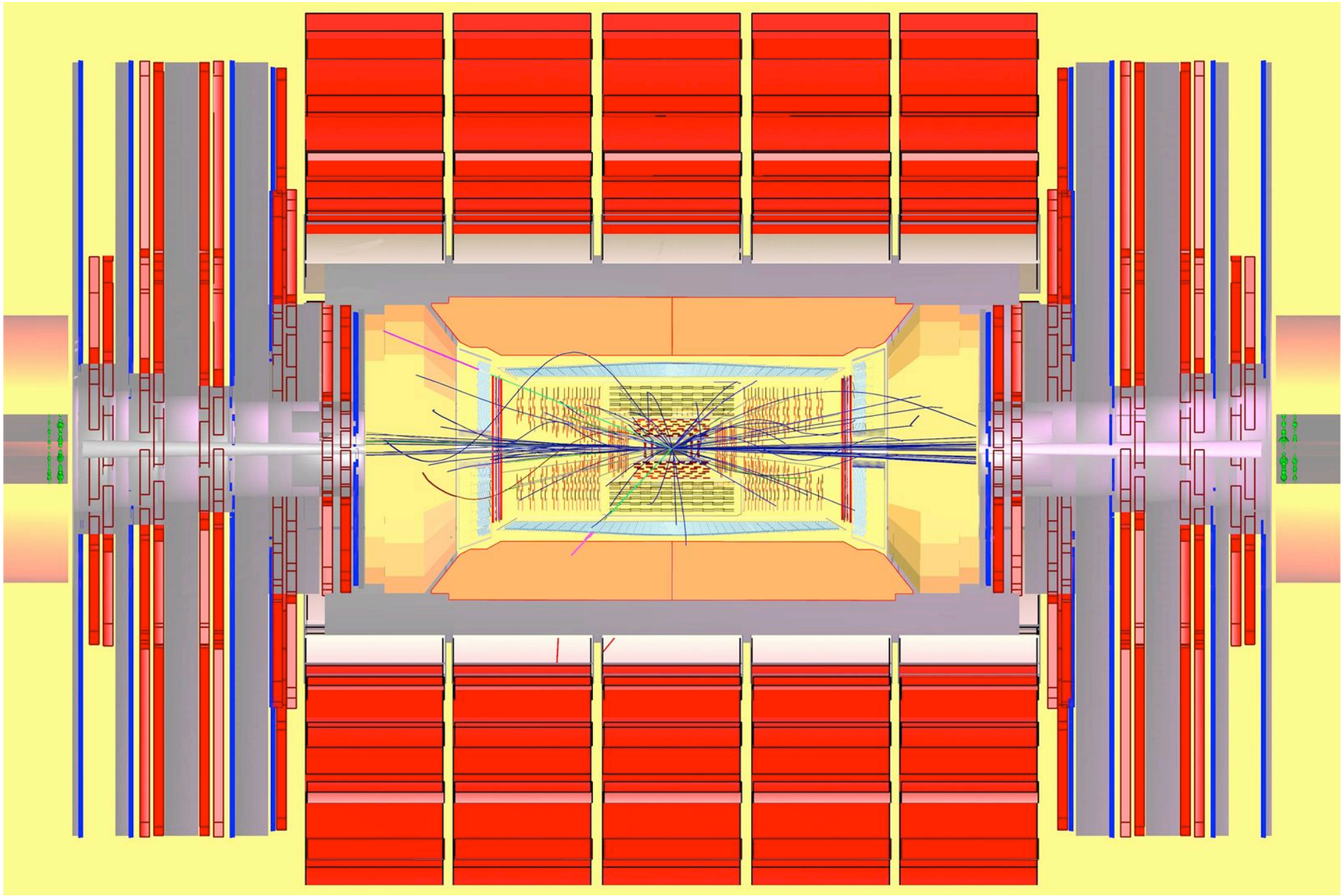
Overview of CMS

Example of inner detector:



The tracker geometry is quite complicated…

[Root graphical version of G4 geometry shown]

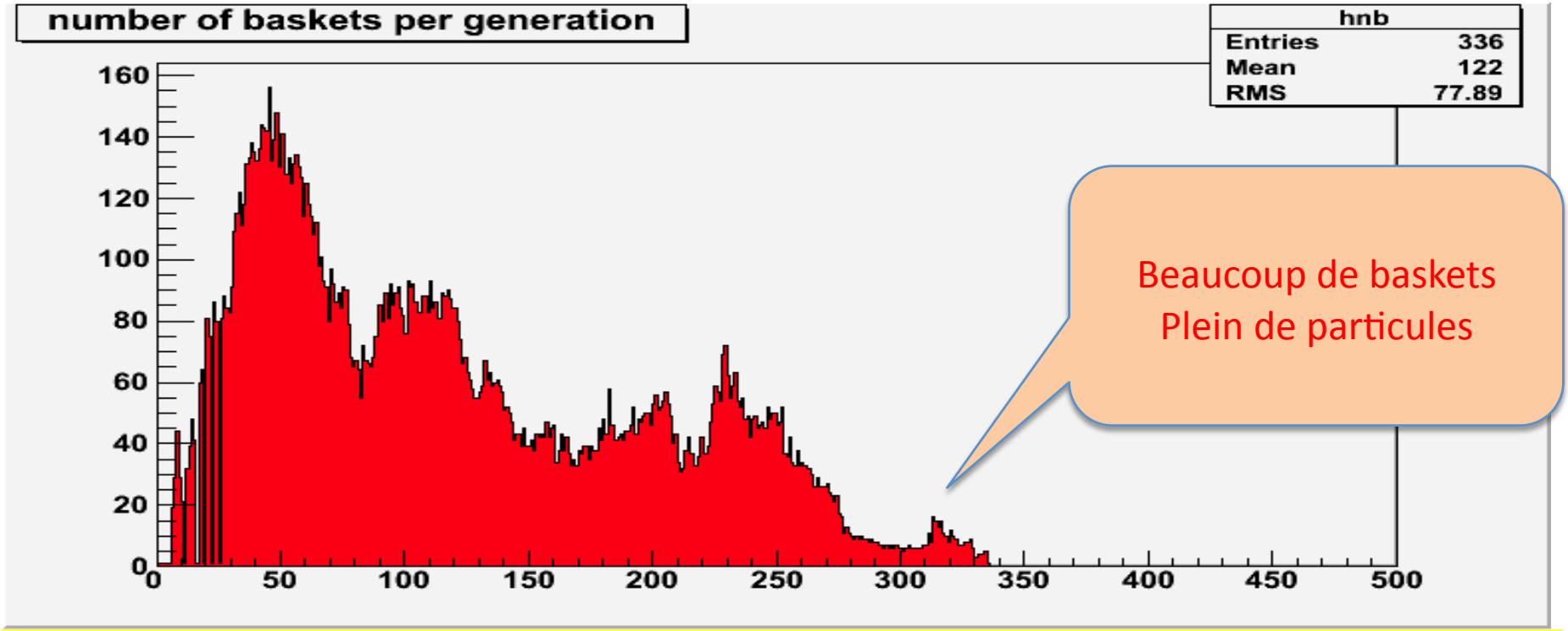


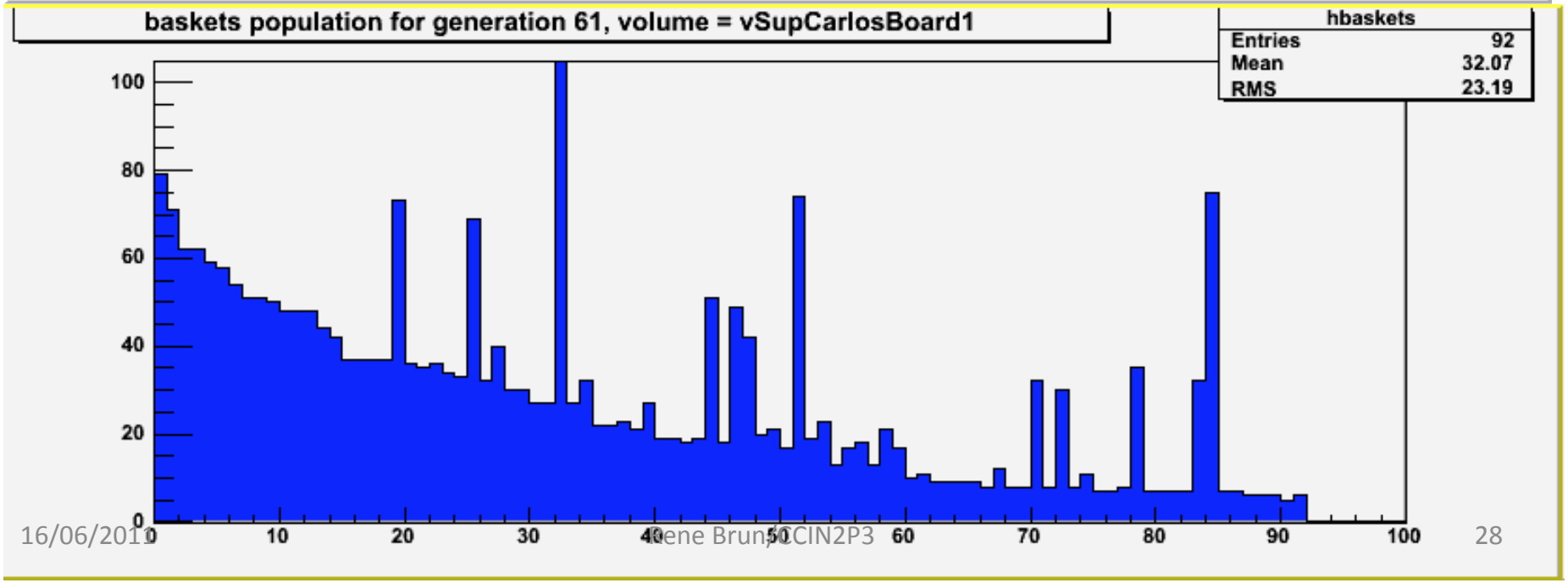
Problèmes avec la méthode actuelle

- Navigation « up/Down » dans une géométrie complexe (typiquement 15 niveaux) avec re-calcul fréquent des matrices. Par exemple ATLAS a 4500 types de volume et 29 millions d'instances.
- Navigation dans des grosses structures de données s'étalant sur des centaines de Mbytes provoquant des « caches misses ».
- Pas de vectorisation possible dans les fonctions de géométrie ou de physique.
  - Distance = DisToOut(volume*, x,y,z,px,py,pz)

Notre prototype

- Les « tracklets » (morceaux de traces dans un même type de volume) sont accumulées dans les volumes de même type (un « basket ») puis transportées en parallèle lorsque le nombre de tracklets dans un basket dépasse une taille critique.
- Afin de générer une taille de vecteurs suffisante plusieurs évènements peuvent être transportés en même temps (nécessaire de toute façon pour les études de pile-up).
  - DistToOut(volume**,n,x*,y*,z*,px*,py*,pz*,dist*)


number of baskets per generation
hnb
Entries 336
Mean 122
RMS 77.89
Beaucoup de baskets
Plein de particules


baskets population for generation 61, volume = vSupCarlosBoard1
hbaskets
Entries 92
Mean 32.07
RMS 23.19

Notre prototype (2)

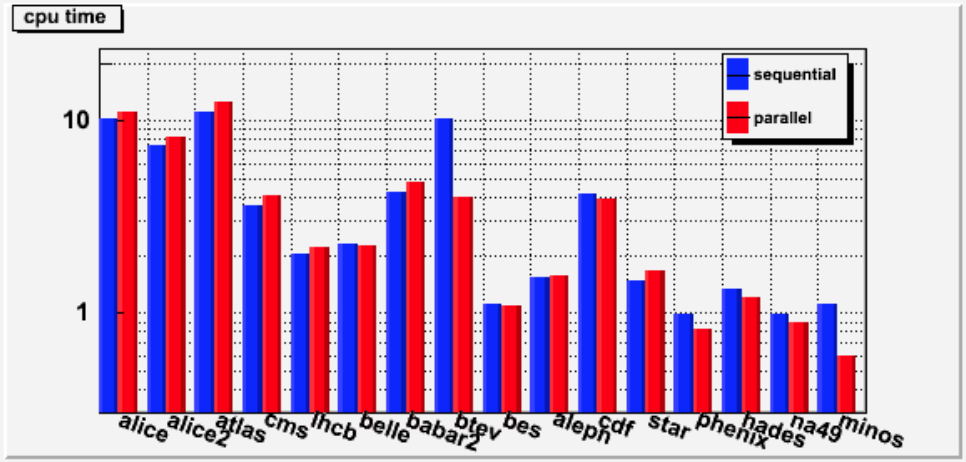
- Est dynamique. Il permet d'adapter le problème: évènement avec peu ou beaucoup de particules, dans un détecteur plus ou moins complexe au nombre de processeurs et architecture (cores et/ou gpus). Le système s'adapte de façon quasi-automatique a la configuration hardware.
- Il permet l'exploitation du pipeline sur un processeur.

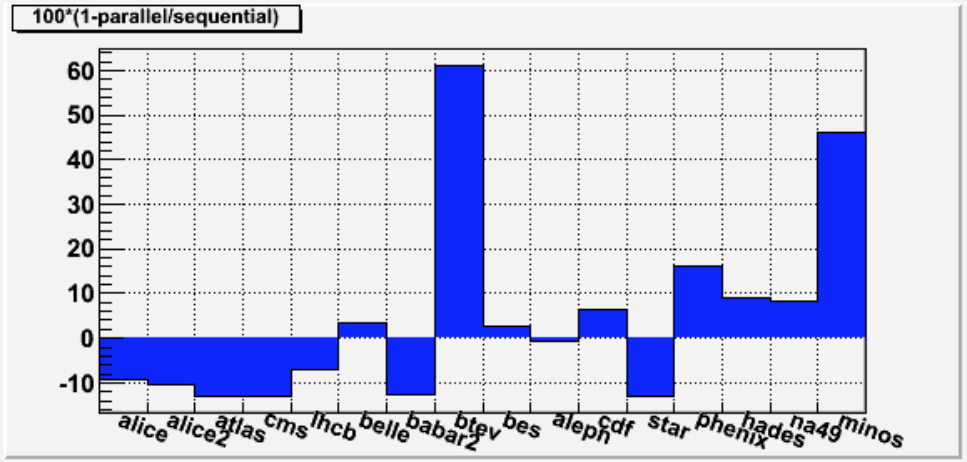
Notre prototype (2)

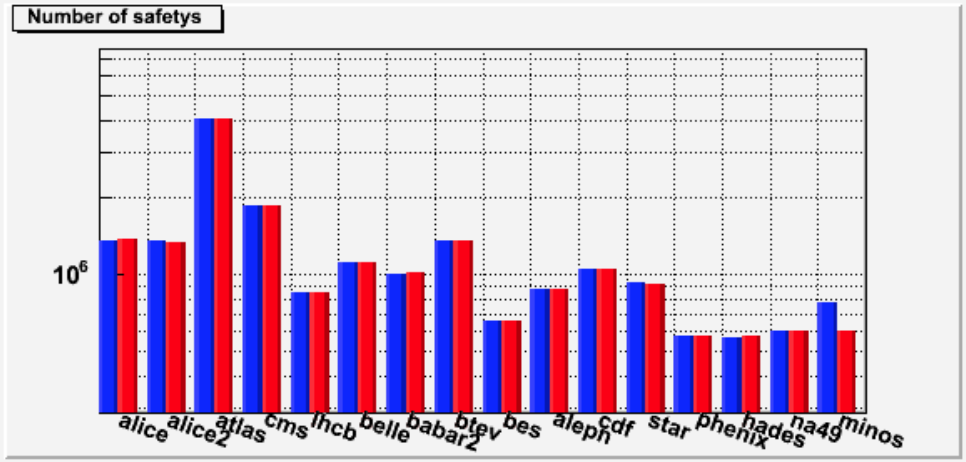
- Les structures de données pour chaque basket sont indépendantes (sauf au top niveau) et l'espace mémoire réutilisable (pools) pour éviter les mutex et la fragmentation.
- Ainsi chaque basket peut être traité par un processeur, ou le vecteur de particules dans un basket traité par un ensemble de gpus.
- Nous investiguons des implémentations pratiques avec OpenMP et CUDA/OpenCL.

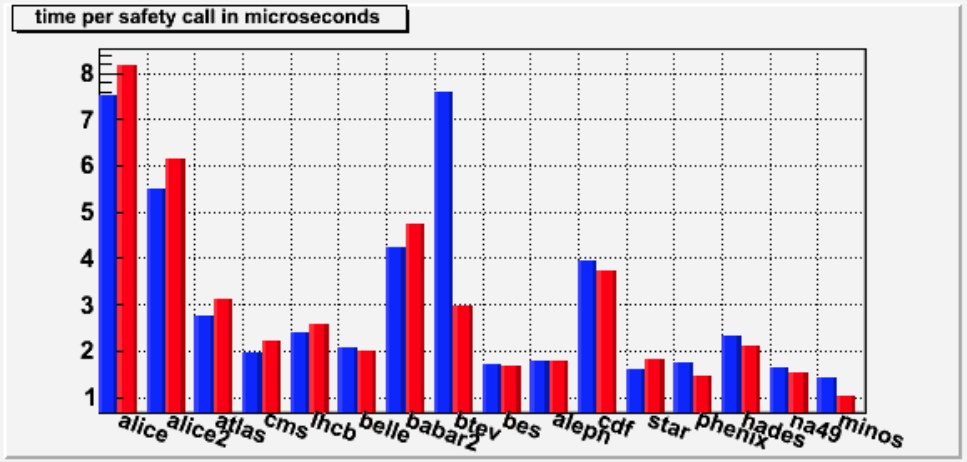
Notre prototype (3)

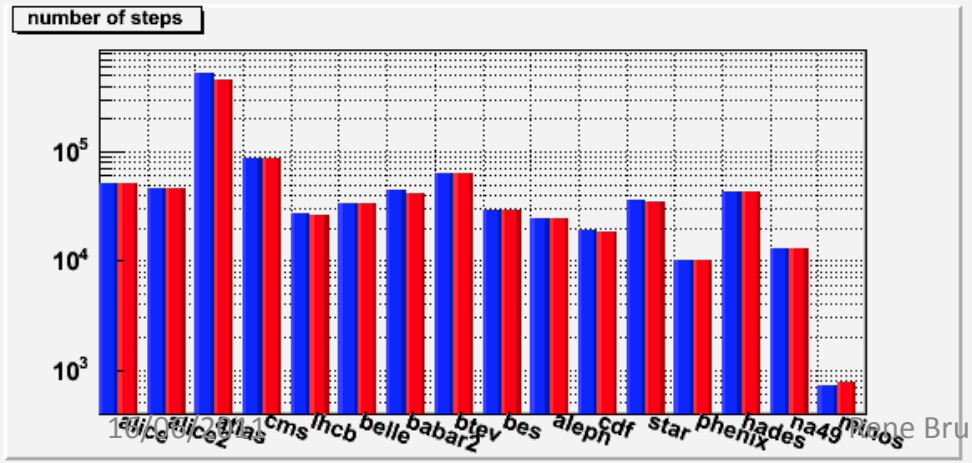
- Le passage du prototype a Geant5 demandera beaucoup de travail (entre 2 et 4 ans)
- Adaptation du nouveau navigateur soit a la géométrie ROOT ou G4
- Vectorisation de la géométrie et de la physique
- Certification
- Implémentation effective sur des systèmes parallèles.

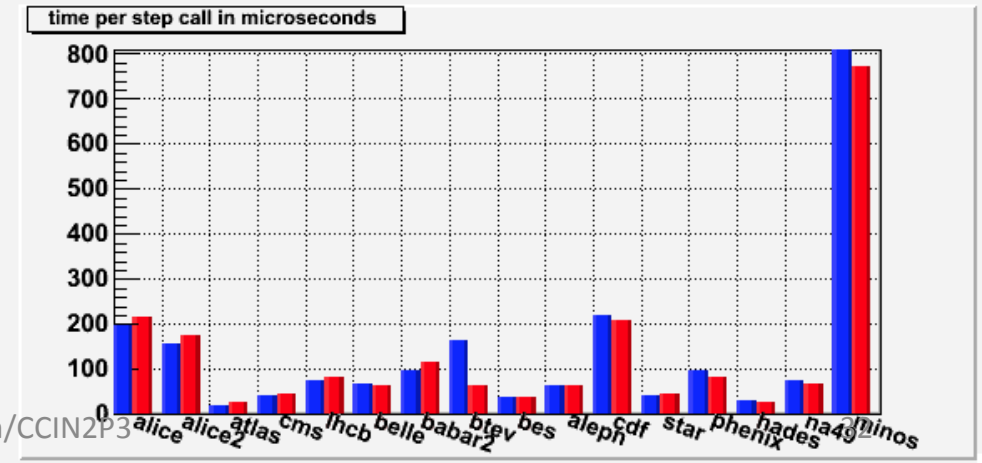

cpu time
sequential
parallel
10
1
alice
alice2
atlas
cms
lhcb
belle
babar2
btev
bes
aleph
cdf
star
phenix
hades
na49
minos


100*(1-parallel/sequential)
60
50
40
30
20
10
0
-10
alice
alice2
atlas
cms
lhcb
belle
babar2
btev
bes
aleph
cdf
star
phenix
hades
na49
minos


Number of safetys
10^6
alice
alice2
atlas
cms
lhcb
belle
babar2
btev
bes
aleph
cdf
star
phenix
hades
na49
minos


time per safety call in microseconds
8
7
6
5
4
3
2
1
alice
alice2
atlas
cms
lhcb
belle
babar2
btev
bes
aleph
cdf
star
phenix
hades
na49
minos


number of steps
10^5
10^4
10^3
alice
alice2
atlas
cms
lhcb
belle
babar2
btev
bes
aleph
cdf
star
phenix
hades
na49
minos


time per step call in microseconds
800
700
600
500
400
300
200
100
0
alice
alice2
atlas
cms
lhcb
belle
babar2
btev
bes
aleph
cdf
star
phenix
hades
na49
minos

Sommaire

- Les logiciels de HEP s'adaptent graduellement a l'environnement hardware (réseau, clusters).
- Un effort particulier est réalisé dans le domaine de la simulation pour être prêt a tirer avantage des systèmes parallèles.
- La réécriture de programmes conçus pour travailler séquentiellement est un gros travail.