dapnia SIS

cea

saclay

Prototype de serveur de monitoring pour le système d'alignement des chambres à Muons (Barrel) de l'expérience Atlas

Andrea Formica, Frederic Chateau

CEA/Saclay DAPNIA/SEDI

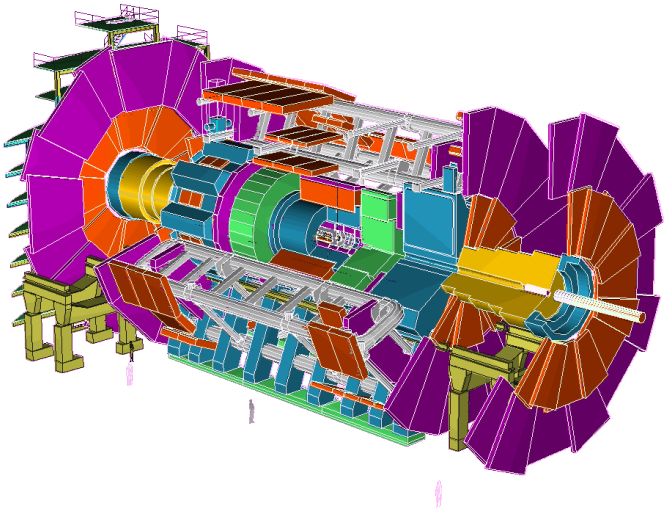
Journées informatique 2006, Lyon

Sommaire

- Détection des muons dans Atlas :
  - Spectromètre à muons et système d'alignment
- Schèma de l'acquisition des données
- Bésoins pour le système de monitoring
- Technologies utilisées
  - J2EE
  - ROOT/C++
  - CORBA
  - Bases des données
- Prototype du serveur d'application pour les services de monitoring de l'alignement et reconstruction de la géométrie
- Autres applications possible pour le même système
  - Le monitoring du champs magnétique d'Atlas (Toroide)

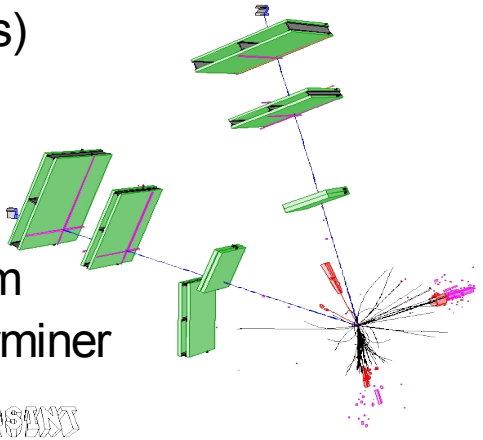
Le spectromètre à Muons dans Atlas

- Détection des désintegrations leptoniques H->4μ
- Dimensions du spectromètre :
  22m haut x 45m de large
- Principe de la mesure: trajectoires des muons courbées par un champ magnétique toroïdal
  - Déterminer la courbure de la trajectoire
  - Mesurer la valeur du champ magnétique
  - Mesurer l'impulsion du muon



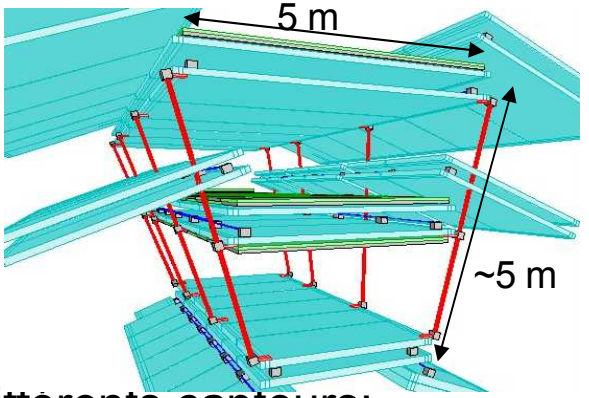
On utilise des chambres à dérive multi-tube pour determiner la trajectoire et on mesure le champ magnétique à travers un système de sondes de Hall placées sur les chambres elles-mêmes (~1700 sondes)

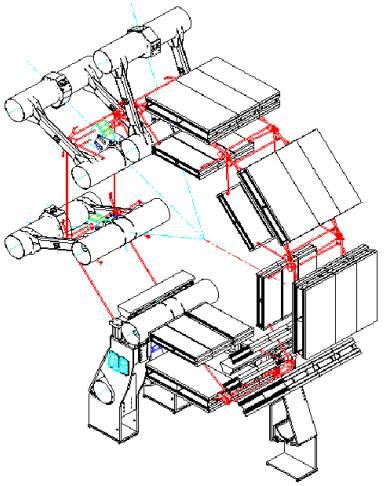
1200 chambres à dérive disposées en trois couches (380000 tubes)
Résolution dans la détermination de la trajectoire ~60μm pour un muon de 1 TeV (courbure de environ 600μm)
Mesure du champ à travers les sondes : résolution ~20 gauss
Mesure de la position géométrique des chambres : résolution 30μm
La connaissance de la géométrie du spectromètre permet de determiner aussi la cartographie du champ magnétique



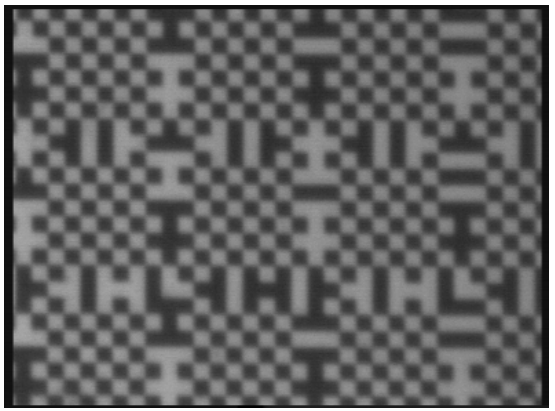
Le système d'alignement dans le Barrel

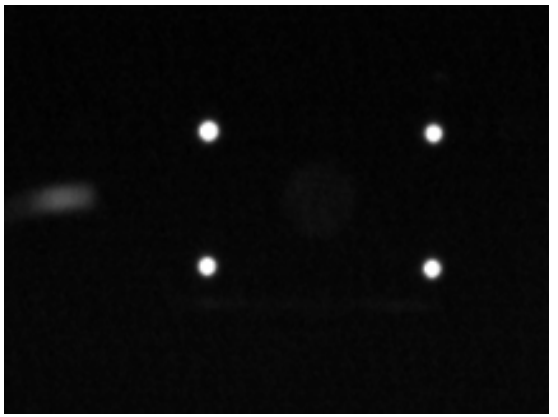
- Positionnement des chambres à muons dans Atlas: précis à ~5mm (ou moins…)
  - Nécessité d'avoir un système d'alignement ayant une gamme de mesure de ±5mm et ±5mrad et une précision de ~30µm et ~200µrad
- Système d'alignement optique basé sur les suivants types de senseurs (~5800):
  - InPlane
  - Axial/Proximité
  - Projectif
  - Référence sur toroïde
  - CCC (chamber to chamber)
  - BIR/BIM



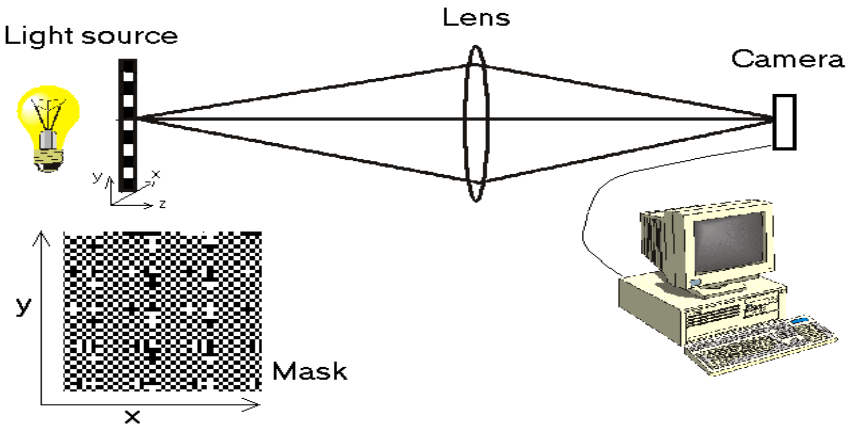


- Deux procédés utilisé pour les différents capteurs:
  - Masque codé (Nikhef)
  - Spots lumineux (Saclay)





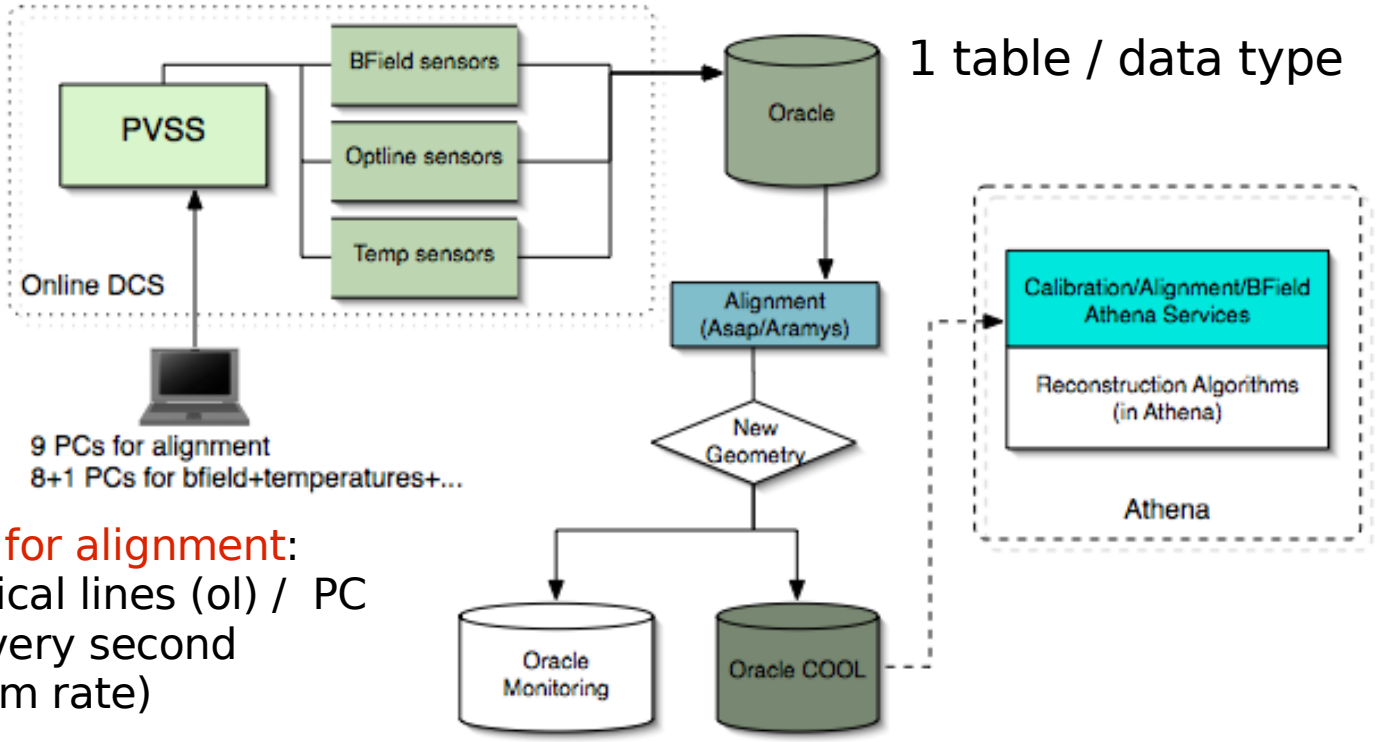
Les systèmes Rasnik et Sacled



- Système Rasnik (masque codé) :
  - Masque (pas de 170 à 600µm);
    Lentille (40<f<1000);
    caméra CMOS (px 12µm)
  - Analyse de l'image acquise
    par la caméra : 4 paramètres
    x,y (<1µm) ; θ (~0.1 mrad) ; G (5x10-4)
- Système Sacled (spots lumineux) :
  - Spot rétro-illuminé;
    CMOS et lentille
  - Analyse de l'image pour la détermination de la position des 4 spots

- Même système d'acquisition pour les 2 systèmes :
  - Capture des images et analyse des données en ligne par le programme d'acquisition (Nikhef)
  - Stockage des paramètres après analyse dans une base de données Oracle via le logiciel PVSS

Schéma d'acquisition



Input data rate for alignment:
9PC , ~600 optical lines (ol) / PC
1 ol read out every second
10 Hz (maximum rate)

Input data size:
1 ol ~ 130 by
total volume in 100 days: 10Gby

Alignment system reading data from Oracle and performing geometry reconstruction
Output data:
set of values (positions + rotations) / chamber (~1500) stored in Oracle monitoring tables and then migrated to COOL (Condition DB)

Offline code will access COOL to retrieve geometry corrections

Tâches du système de monitorage de l'alignement

dapnia SIS

cea

saclay

- Le système de monitoring de l'alignement doit vérifier les données des lignes optiques à intervals regulier pour determiner s'il y a besoin ou pas de lancer une reconstruction géométrique sur un certain interval de temps
- Le programme qui effectue cette reconstruction (ASAP) est implementé en C++ et utilise des fonctionnalités ROOT
- Les corrections géometriques doivent être enregistrées dans des tables sur une base de données de monitoring (sous Oracle), pour être après validation migrée dans la base des données Condition DB (toujours Oracle).

- Les suivantes fonctionnalités sont demandées :
  - Accèder à plusieures tables dans une (ou plusieures) base Oracle (ou autre)
  - Configurer la frequence de vérification des valeurs des lignes optiques
  - Configurer la géométrie en entrée du programme d'alignement
  - Lancer le programme d'alignement sur un interval de temps
  - Gérer les corrections géométriques en sortie
  - Accèder par des programmes clients aux info de monitoring
  - Prévoir un accès par client web aux fonctionnalités du monitoring

Services de monitoring pour l'alignement

- IntervalMaker : ce service consulte avec une certaine fréquence les données des lignes optiques pour établir s'il faut ou pas reconstruire la géométrie
- AsapService : ce service est censé lancer l'application Asap (C++) sur un lot de données vénant des lignes optiques, et avec une certaine configuration géométrique nominale (chambres on/off, lignes optiques on/off, ...)
- ReconstructionScheduler : ce service comunique avec IntervalMaker pour savoir s'il faut faire une reconstruction, et en cas affirmatif declanche la reconstruction à travers le AsapService
- Algorithme d'alignement utilisé : librairie C++ (basé sur le framework ROOT)
  - Input:
    - géométrie nominale
    - Mesures des lignes optiques dans un certain interval de temps
  - Output:
    - corrections géométriques
    - X^2 du fit et autres paramètres (nombres de senseurs utilisés, etc.)

Technologies utilisées

- C++ / ROOT : fit d'alignement
  - L'utilisation de ROOT paraît naturel dans le contexte hep
- Java 1.5 / J2EE (EJB3) : fonctionnalités de serveur, connections aux bases des données et persistance d'objets java dans le monde relationnel
  - Nous avons choisie une des technologies les plus répandues dans l'industrie informatique (EJB3 sont très nouveaux)
  - L'implémentation utilisée du serveur d'application est JBOSS (Open Source)
    - actuellement une des implémentations les plus avancés le team JBOSS étant parmi les plus actifs dans la définition des spécifications EJB3 (basée sur les mécanismes de persistance de Hibernate).
- Bases des données (Oracle / MySQL ...) et XML
  - Oracle est choisi par la communauté Atlas et CERN en général, la division IT du CERN ayant une équipe d'administration Oracle très forte.
  - XML : utilisé pour les descriptions de géométrie en C++ , et pour tout fichier de description en Java (spec. J2EE)
- CORBA
  - Ce protocole a été utilisé pour la communication entre les programmes Java et C++, pour séparer les 2 mondes d'une manière propre.
  - Une première utilisation des librairies JNI (java native interface) a échoué à cause d'un plantage de la machine virtuelle Java suite au simple loading d'une librairie dynamique ROOT

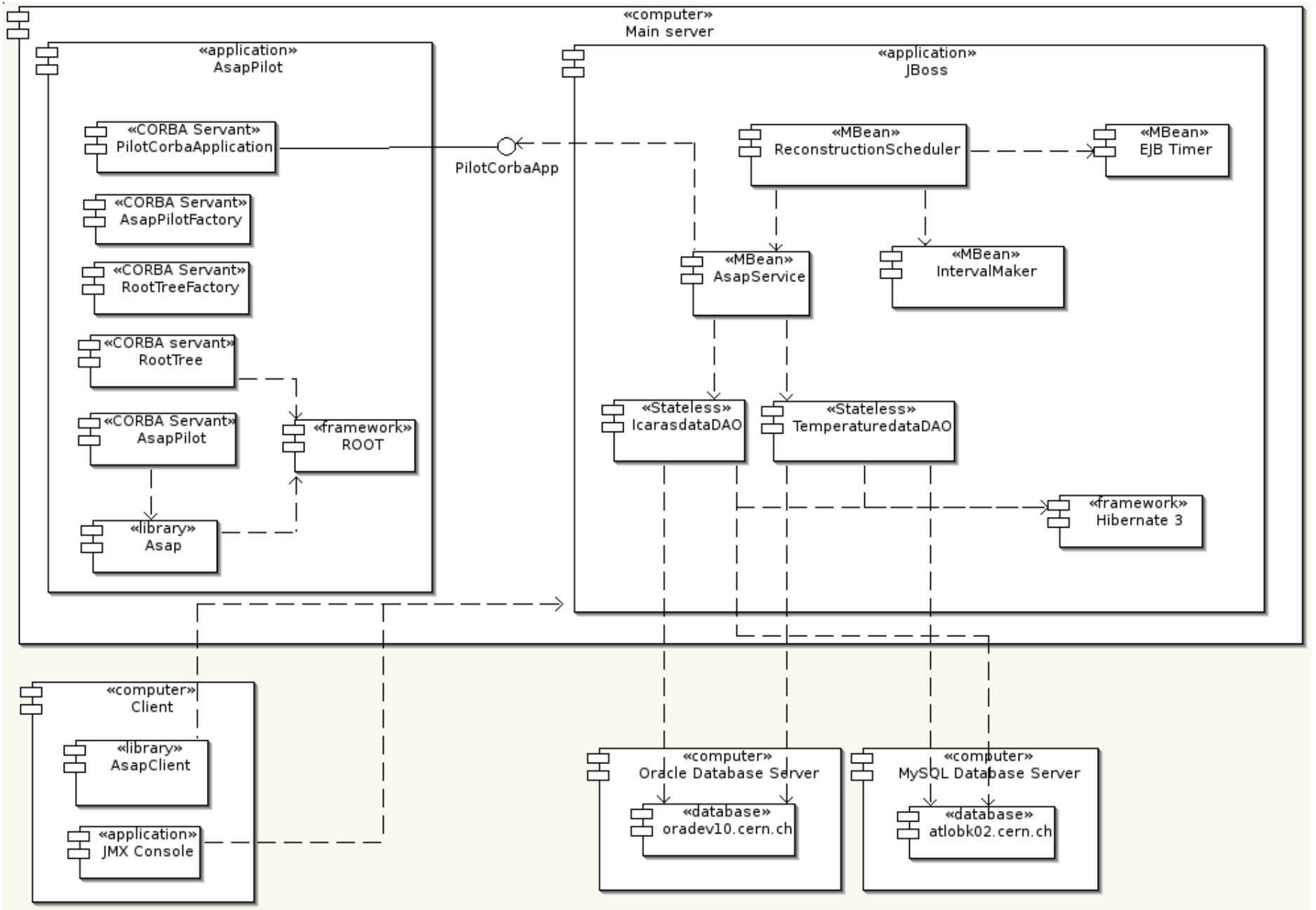
Bref description de l'environnement J2EE

- Avantages de J2EE :
  - J2EE permet de séparer la partie métier d'une application et la partie cliente: il s'agit d'un modèle multi-tier dans lequel figurent des bases des données pour le stockage, un serveur d'application pour la partie métier, et une librairie cliente (ou Web) plus légère.
  - Fonctionnalités de serveur intégrées
  - Gestion des connexions aus BDD à travers des Pool complètement configurables à partir de simple fichiers XML (nombre de connexions au démarrage, max nombre des connexions, ...)
  - Gestion du cycle de vie des objets, des transactions et de la mémoire
  - Intégration simple avec des services WEB (pages JSP, Servlets, SOAP)
- EJB3 vs EJB2:
  - Les EJB3 permettent de mapper dans une base des données relationnelle une arborescence de simple objets Java (POJO)
    - On utilise pour ça un système d'annotations (sorte de commentaires qui sont interprétés au moment de la compilation)
  - L'utilisation de POJO permet un développement beaucoup plus rapide par rapport aux EJB2 (basés eux sur l'implémentation d'interface), et en plus simplifie la librairie cliente (puisque les POJOs peuvent exister aussi dans un programme Java normale, non intégré à un Serveur d'Application)

Description des EJB3

- Trois différent types de Beans sont disponibles (suivant les fonctionnalités dont on a besoin):
  - EntityBeans : il s'agit des objets qu'on veut rendre persistants, et donc des données de notre application
  - SessionBeans (Stateless / Statefull): il s'agit des objets qui contiennent l'algorithme de l'application (ils opèrent avec les EntityBeans). Ils peuvent garder en mémoire l'état associé à un client particulier (Statefull). Les appels par plusieurs clients du même SessionBean peuvent être parallélisés par la création de plusieurs Thread d'exécution.
  - MessageDrivenBeans : il s'agit d'objets qui peuvent gérer des files d'événements, et donc d'effectuer des actions d'une manière asynchrone
- Utilisation des annotations :
  - A partir de Java1.5 les annotations sont disponibles dans les librairies java
  - Ex d'annotation pour un EntityBean :
    - Annotation pour une classe: @Entity, @Table(name="MYTABLE"),...
    - Annotation pour un attribut : @Column(name="xValue", precision=15)
private BigDecimal xvalue;
  - Ex d'annotation pour un SessionBean :
    - Annotation pour une classe : @Stateless, @Remote(myclass.class)
    - Annotation pour un attribut :
@PersistenceContext(unitName="mycontext")

Schéma de l'application de monitoring



JMX console management interface / 1

JMX MBean View

MBean Name: **Domain Name:** AsapServer
service: AsapService
MBean Java Class: asap.server.asap.AsapServiceMBean

Back to Agent View Refresh MBean View

MBean description:

Information on the management interface of the MBean

List of MBean attributes:

Name	Type	Access	Value	Description
StateClass	java.lang.Class	R	class asap.server.asap.states.AsapServiceReady	Attribute exposed for management
ConfigFilePath	java.lang.String	R	/home/fchateau/asap/Asap2/config/config_2004.xml	Attribute exposed for management
Since	java.util.Date	R	ven., 20 août 2004 08:00:00 +0000	Attribute exposed for management
Till	java.util.Date	R	ven., 20 août 2004 10:00:00 +0000	Attribute exposed for management
IterationCount	java.lang.Integer	R	5	Attribute exposed for management
FitMode	java.lang.Integer	R	0	Attribute exposed for management
Verbosity	java.lang.Integer	R	0	Attribute exposed for management
IcarasMeasuresCount	java.lang.Integer	R	1741	Attribute exposed for management
TemperatureMeasuresCount	java.lang.Integer	R	0	Attribute exposed for management
StateTitle	java.lang.String	R	Ready	Attribute exposed for management
StateDescription	java.lang.String	R	Asap reconstruction ready to start.	Attribute exposed for management

List of MBean operations:

void start()

Operation exposed for management

Invoke

JMX console management interface / 2

dapnia SIS

cea

saclay

JMX MBean View

MBean Name: **Domain Name:** AsapServer
service: ReconstructionScheduler
MBean Java Class: asap.server.scheduler.ReconstructionSchedulerMBean

Back to Agent View Refresh MBean View

MBean description:

Information on the management interface of the MBean

List of MBean attributes:

Name	Type	Access	Value	Description
AsapConfigFilePath	java.lang.String	RW	/home/fchateau/asap/Asap	Attribute exposed for management
ExecutionStartTime	java.util.Date	RW	mar., 29 août 2006 12:08::	Attribute exposed for management
ExecutionStopTime	java.util.Date	RW		Attribute exposed for management
ExecutionPeriodicityMinutes	java.lang.Integer	RW	10	Attribute exposed for management
SampleLengthMinutes	java.lang.Integer	RW	30	Attribute exposed for management

Apply Changes

JMX console management interface / 3

dapnia SIS

cea

saclay

- Les services sont implémentés dans notre application de monitoring comme des MBeans.
  - MBeans : sont des classes qui implémentent une interface de management bien défini par les spécifications JMX.
  - Les MBeans s'enregistrent au près d'un MBeanServer et peuvent communiquer entre eux en utilisant le buffer JMX.
  - Des mécanismes de notification peuvent être mis en place pour déclencher des actions par un MBean suite à l'action d'un autre MBean
- L'interface HTML de management des services est automatiquement générée par le serveur d'application sur la base de l'instrumentation du MBean

- L'administrateur peut se connecter à la console JMX de JBOSS et activer ou éteindre des services, les configurer, appeler des méthodes et setter des paramètres, etc....
- En plus des MBeans que nous avons nous mêmes créés notre application utilise le TimerMBean:
  - Ce service disponible dans les librairies Java qu'on utilise, permet de générer des événements d'horloge et de envoyer de notification aux autres services (qui les demandent) pour declancher des actions à intervals régulier.

Conclusions

- Le premier prototype de l'application de monitoring est prêt pour fonctionner avec les données de tests qui pourront être prises pendant la phase de commissioning
- Beaucoup de développement reste à faire :
  - Algorithme
  - Corrections géométriques en sortie
  - Définition du service IntervalMaker
- L'utilisation de CORBA nous paraît la seule solution solide à terme pour la cohabitation des 2 environnements : Java / ROOT(C++)
- D'autres applications de monitoring comme celle du champs magnétique dans Atlas Toroide pourraient bénéficier des développements qui ont étés effectués au sein de l'alignement Barrel.

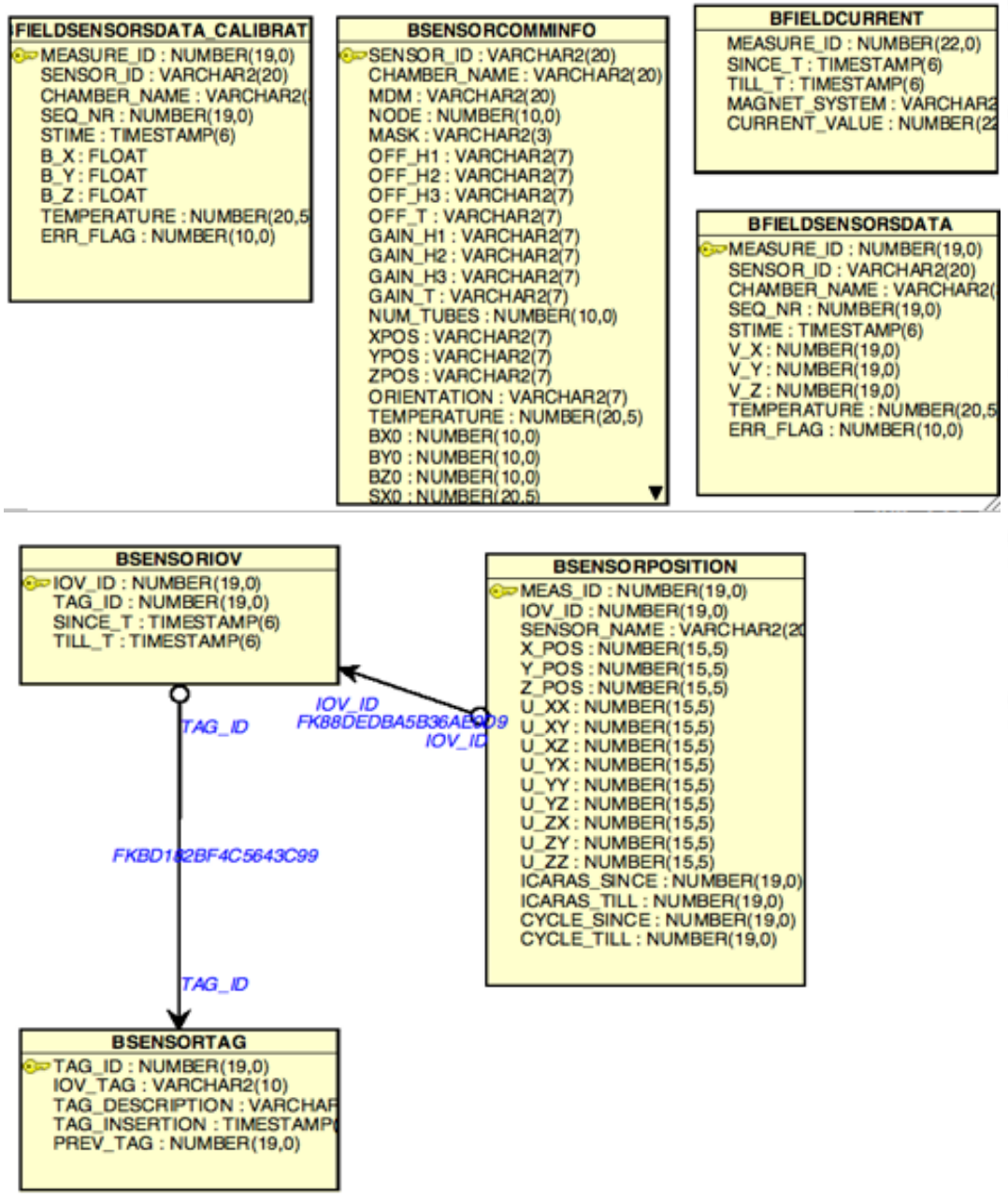
Bfield Monitoring and Oracle table structure

- Tables created in Oracle for Bfield monitoring :
  - DCS data :
    - Raw sensor data : milliVolt measurements
  - Calibrated data :
    - Calibrated sensor data : gauss measurements (Felix Bergsma)
  - Geometry related data :
    - B sensor commissioning information
    - Rotation matrix and sensor position in Atlas reference frame

- The table with largest impact in data size are DCS data and calibrated data

- To perform analysis on bfield sensor data we have choosen to produce a ROOT tree which is then processed by a monitoring program
  - Java program access DB and send data to a tree producer program based on CORBA.
  - A ROOT based program then analyses the data to produce a set of standard histograms

Oracle table schema



- DCS data table is filled by the acquisition program via PVSS
- Calibrated data table is filled using a Java program interfaced with the calibration ascii file (and relative code) provided by Felix Bergsma.
- Commissioning information is used to get the «Asap Name» of a sensor from his sensor_Id
- Bfield Current table is filled from an ascii file retrieved from web service
- A simple mechanism of time varying condition is set up to follow nominal geometry and its variations. The geometry position of a sensor is filled via an ascii file provided by the Asap alignment program