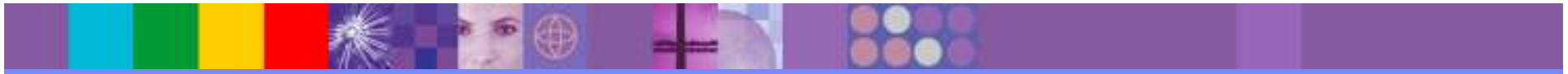
IBM Software Group

WebSphere XD

Compute Grid

An IBM Proof of Technology

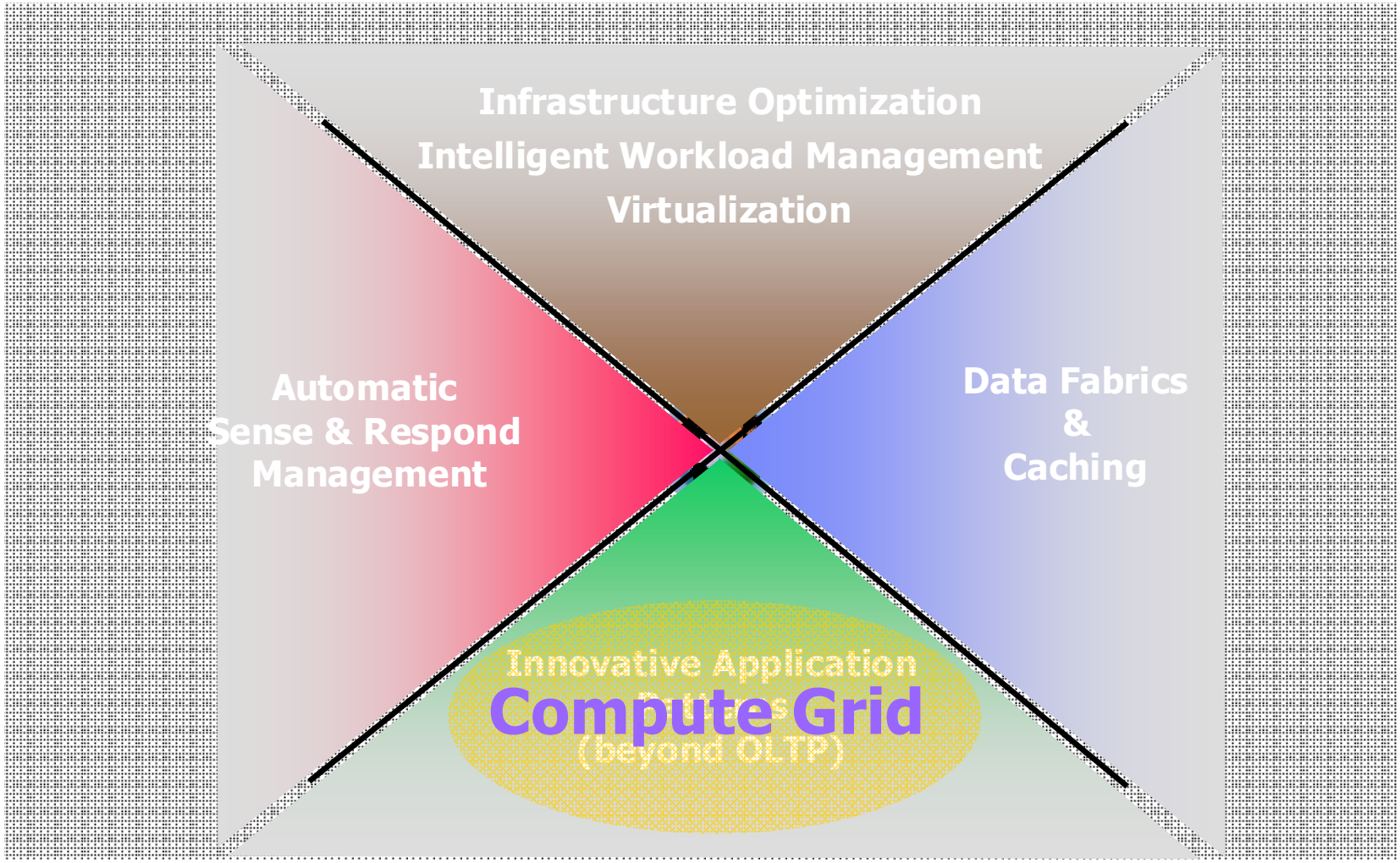


Agenda

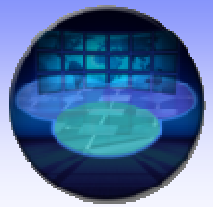
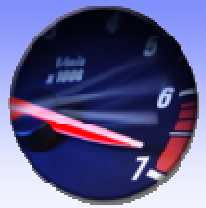
- Introduction
- 3 types de traitement
  - Batch Transactionnel
  - Calcul intensif
  - Natif
- Environment

What is WebSphere XD?

Software to virtualize, control, and turbo-charge your application infrastructure



Application Infrastructure Portfolio

Adoption Patterns		*IBM Offerings*
Application Foundation		WebSphere Application Server Family WebSphere sMash IBM CICS® Transaction Server
Intelligent Management		IBM workload Deployer WebSphere Virtual Enterprise
Extreme Transaction Processing		WebSphere eXtreme Scale WebSphere Real Time WebSphere Compute Grid

Les besoins

- Des processus long, gourmande en CPU, application non traditionnelle type Web/transactionnelle
- Des batchs de plusieurs heures/jours traitant de gros volumes de données
- Application native avec un besoin d'infrastructure administrative
- Faire cohabiter des applications batch avec des applications Web
- L'offre => WebSphere Compute Grid

Les besoins (…)

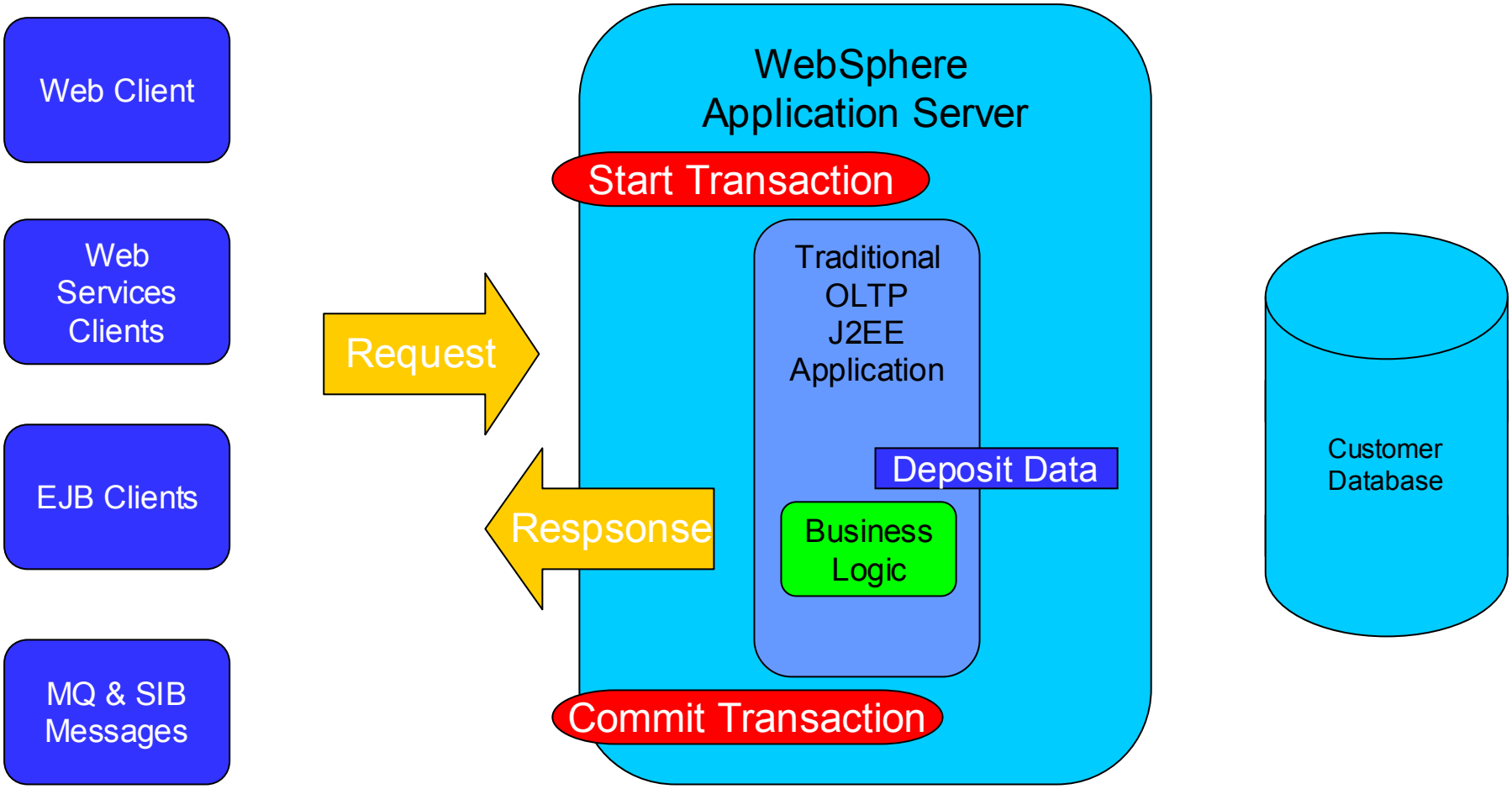
- Les protocoles et les modèles de programmation indépendant de toutes expirations de temps => besoin de persistence
- L'administrateur doit pouvoir superviser, et exploiter les traitements :
  - La soumission/exécution doit être assynchrone
  - Séparer les rôles de celui qui soumet de celui qui exécute
- Exécuter des application non JEE

Compute Grid : les différents types

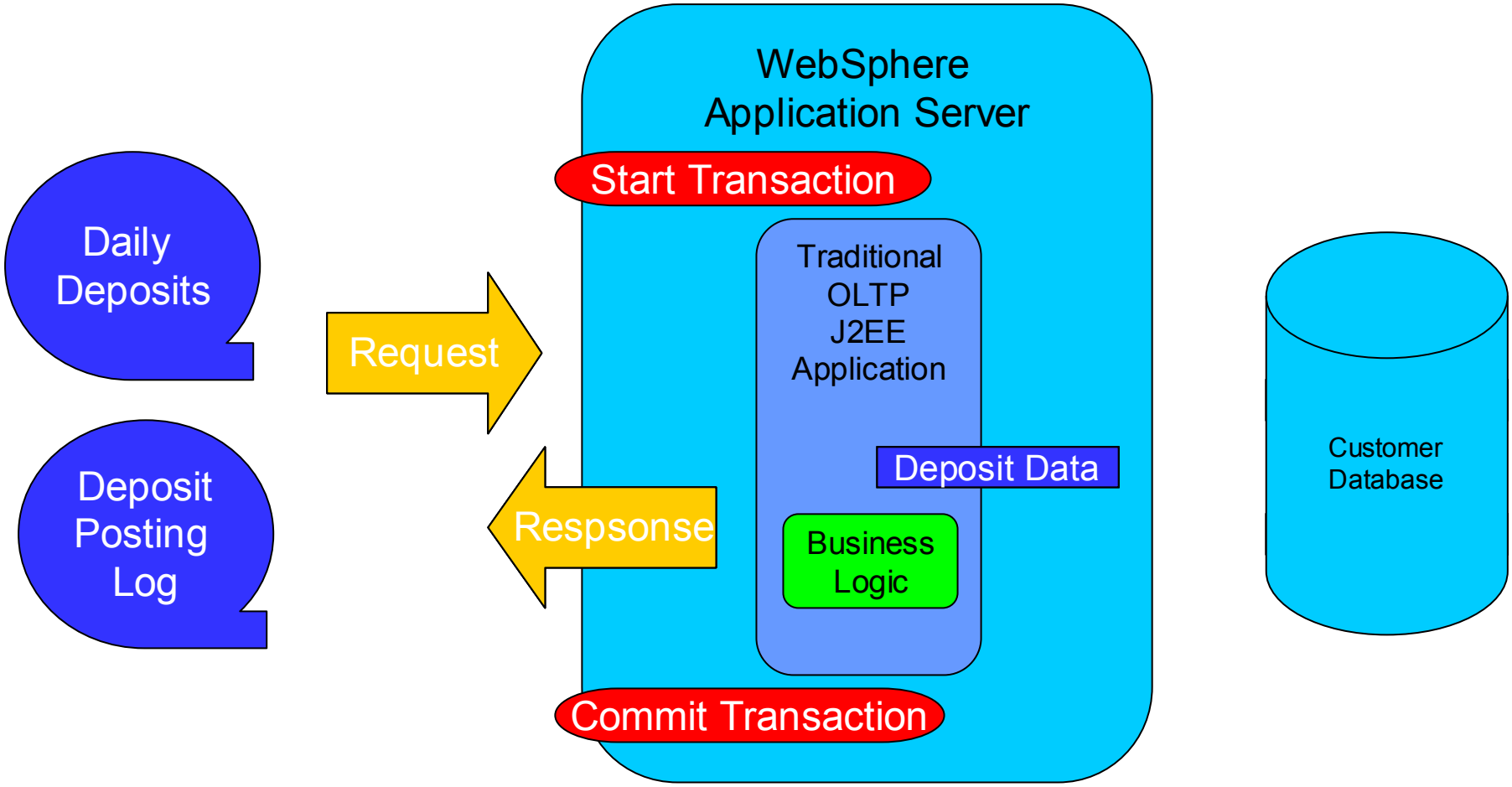
Supporte 3 modèles :

- Batch – une approche à l'enregistrement
  - Le container gère les transactions avec des mécanismes de checkpoint/restart
  - Besoin juste de développer la logique d'un traitement au niveau de l'application
- Calcul intensif
  - Le container lance des threads d'éxécution en parallèle et en remote
  - L'application apporte la logique métier de l'exécution
  - Le container n'interfère pas avec lors de l'exécution des threads de traitement
- Natif
  - en Java, en langages compilés, comme C++, COBOL, et scripts.
  - Pas de contrainte de programmation applicative

Requête traditionnelle – application Web



Gestion des transactions dans une application Batch



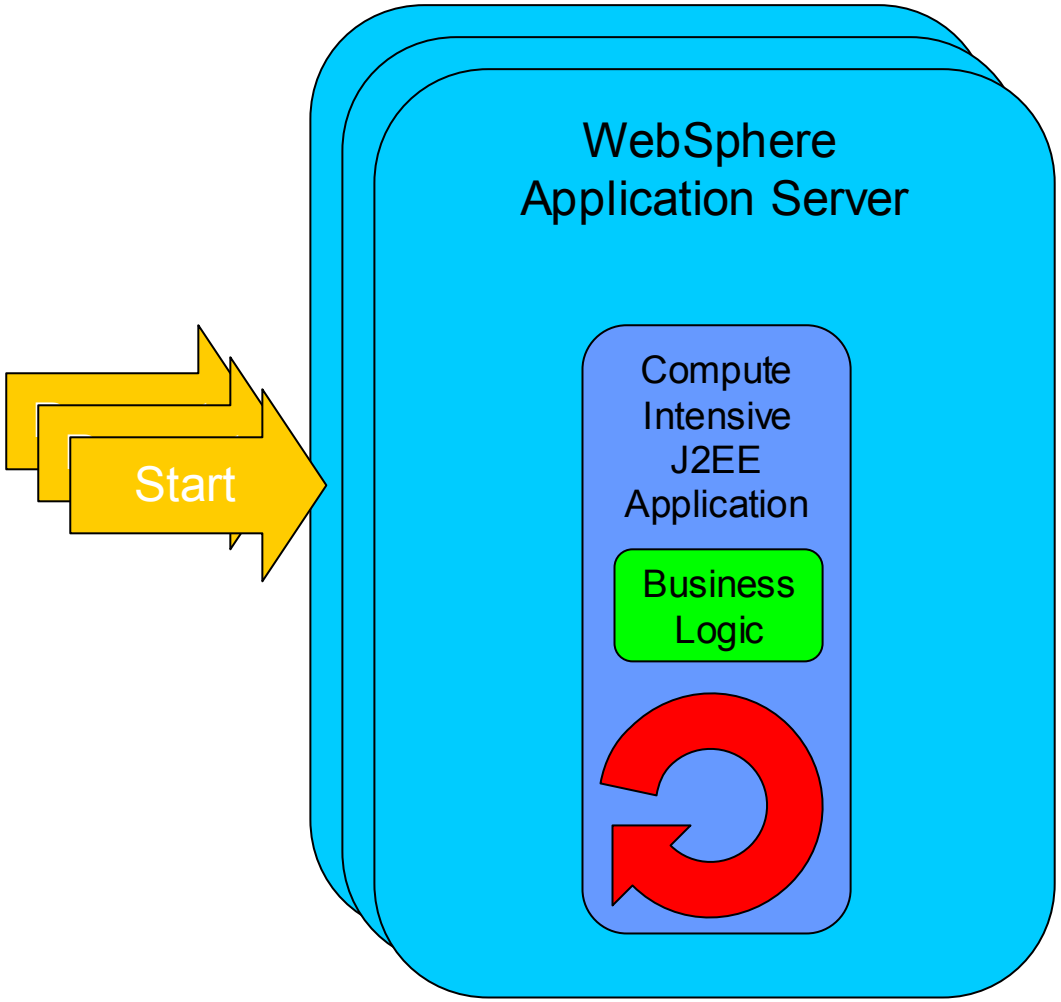
Optimiser le traitement de grands volumes de donnée

- Des opérations répétitives
  - Unité d'œuvre transactionnelle s'étend sur un ensemble d'enregistrement
- Des flux entrant et sortant
  - En entrée : les données à traiter
  - En sortie : les données résultant
- Support des checkpoints
  - A tout moment le traitement peut reprendre à la localisation du dernier checkpoint (curseur)

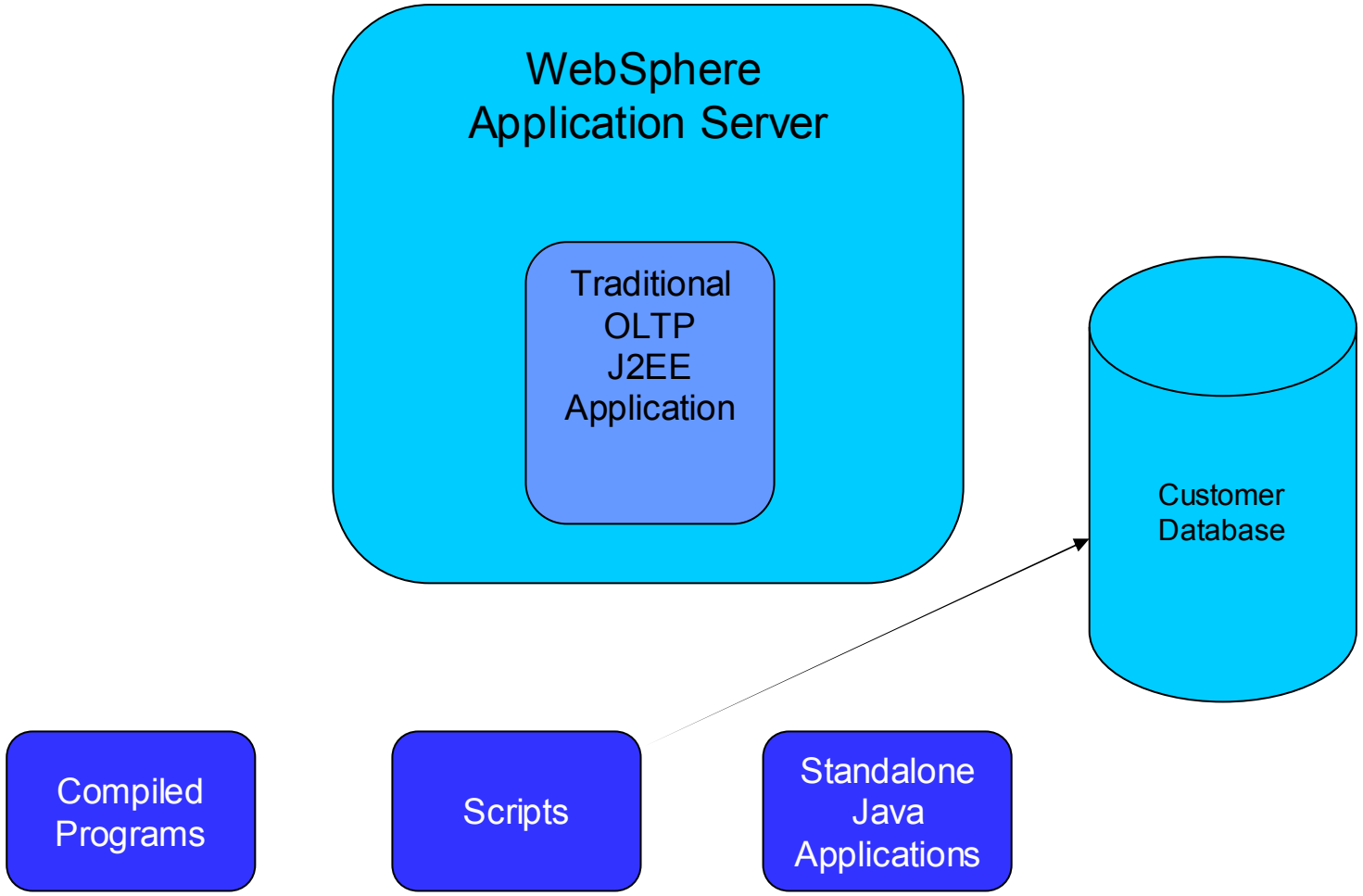
Calcul intensif

- Partitionner le travail
- Exécuter chaque portion en parallèle sur un pool de serveur
- La durée Totale du traitement = au traitement de la partition la plus longue
  - Peut-être écourtée via une gestion de time-out

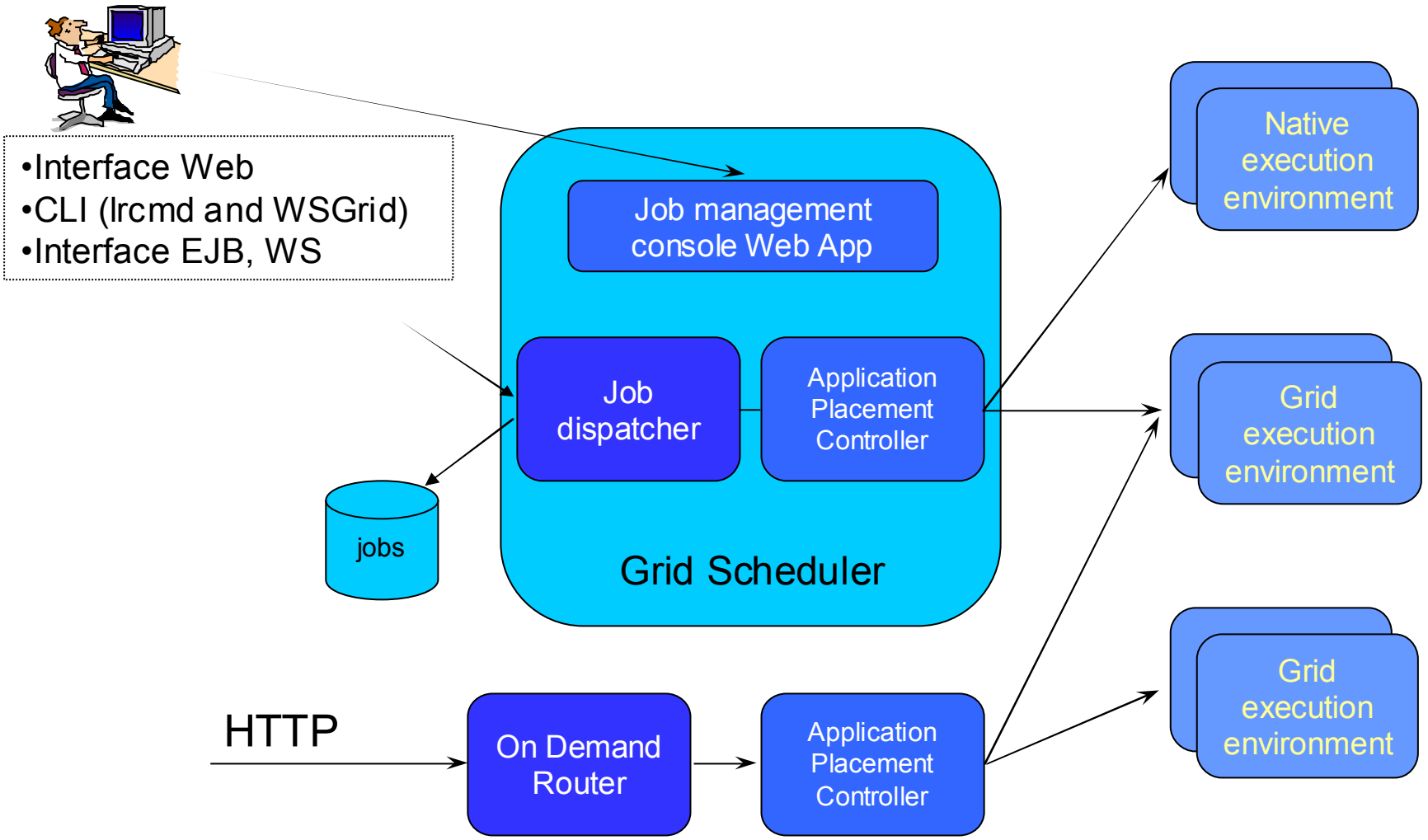
Calcul intensif



Application natives



Architecture



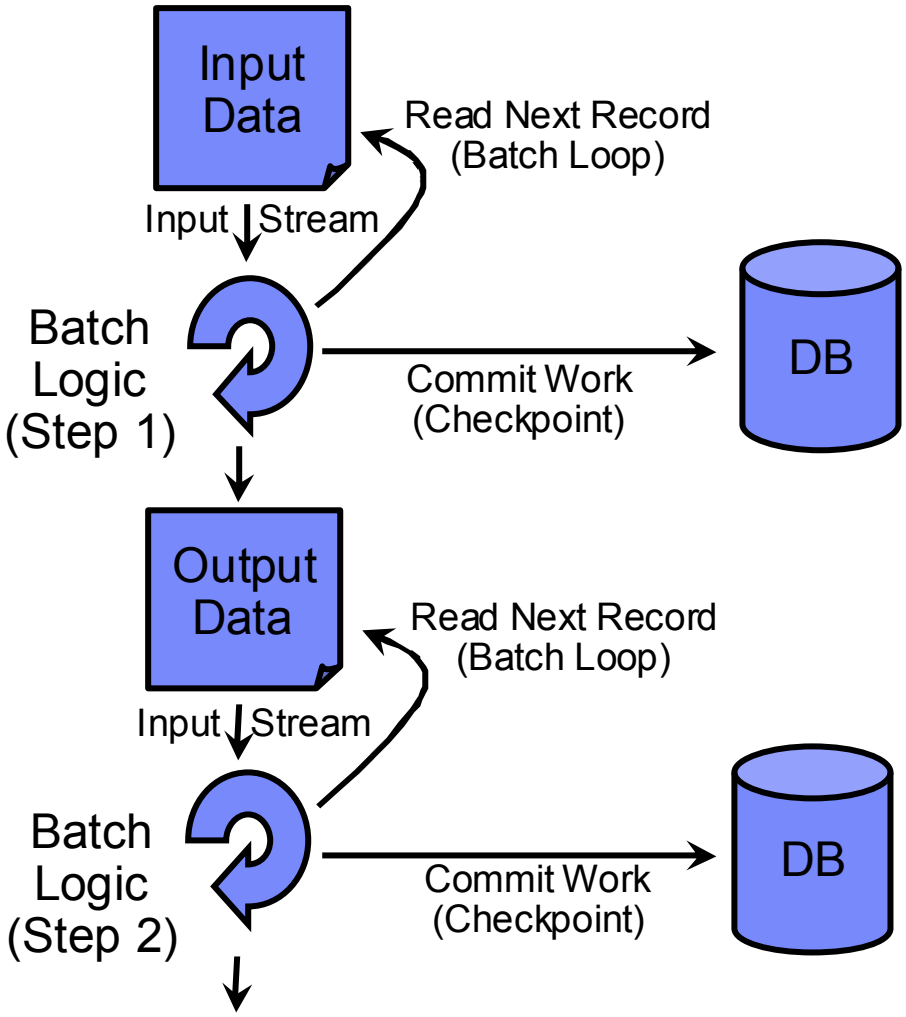
Agenda

- Introduction
- 3 types de traitement
  - Batch Transactionnel
  - Calcul intensif
  - Natif
- Environment

Gestions des batchs dans WebSphere

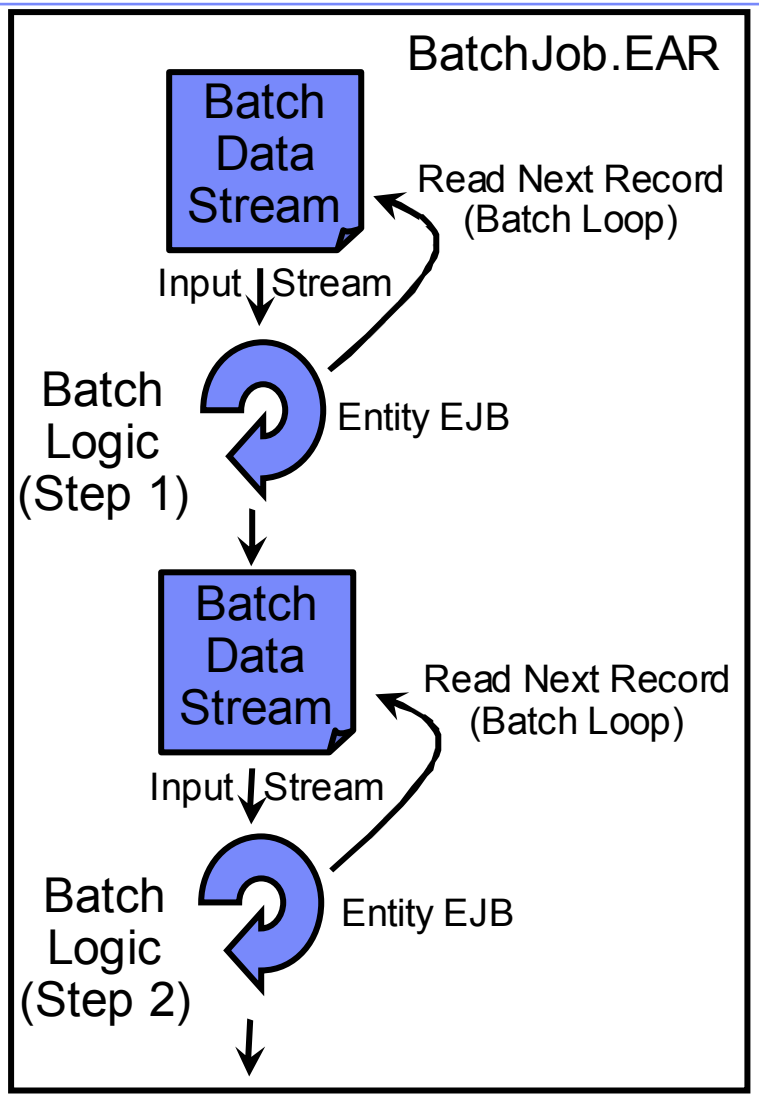
- Basé sur un modèle de programmation JEE
  - Cacul intensif
  - Batchs transactionnels
- Peut inclure l'exécution de service existant (SOA)
- Format xJCL pour décrire l'exécution d'un programme Batch

L'Exécution



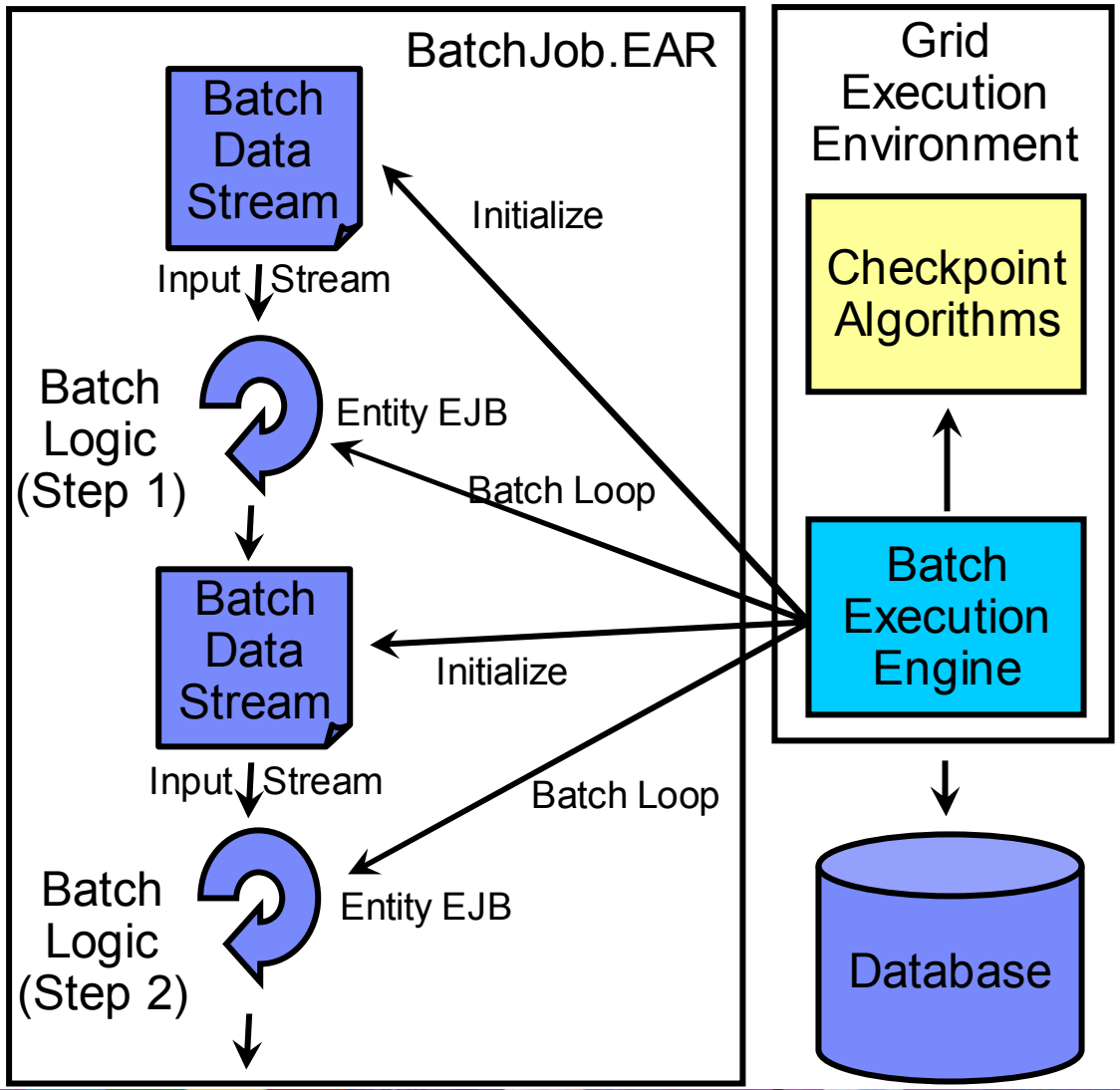
- Processus batch typique
  - Lire un enregistrement de la source de donnée
  - Exécution de la logique métier
  - Ecriture en sortie du résultat
  - Commit de l'unité d'œuvre (checkpoint)
  - Boucler sur l'enregistrement suivant
- Chaque Batch peut être composé de plusieurs traitements (step)
- La sortie d'un traitement est l'entrée du traitement suivant

Modèle de programmation JEE



- Les flux d'entrée et sortie sont encapsulés dans un objet Batch Data Stream (BDS)
- La logique métier se limitte au développement d'un POJO, pas d'EJB, WS, …

L'environnement d'exécution



- Le conteneur (Grid Execution Environment - GEE) initialise les sources de donnée (BDS), invoque en callback la méthode d'exécution de la logique métier et continue la boucle.
- GEE gère la transaction globale tout en exécutant les traitements

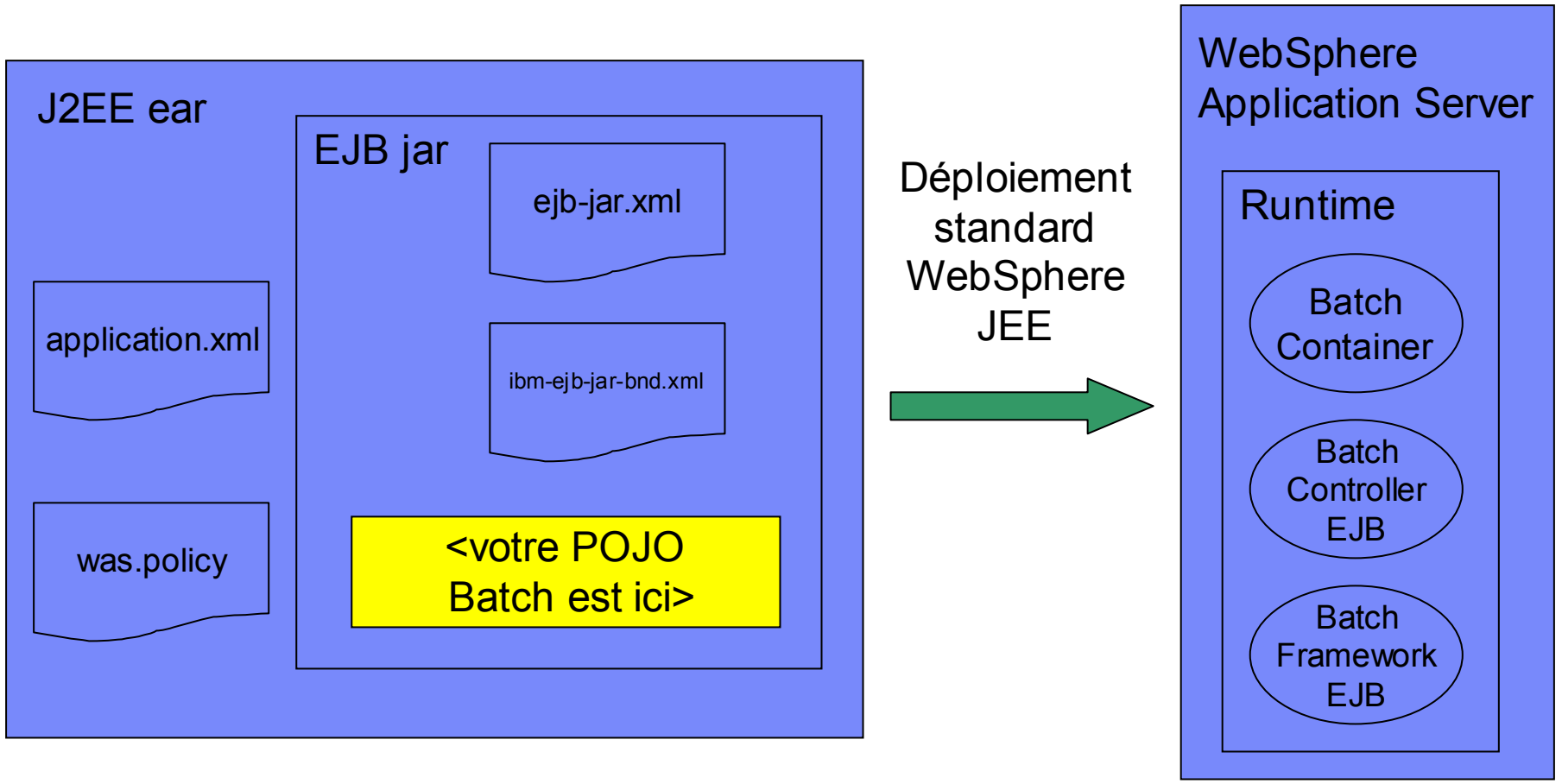
Algorithmes de checkpoint

- Les algorithmes de checkpoints gère le cycle de vie des transactions globales démarées par le conteneur batch (GEE)
  - Jusqu'au commit, GEE récupère la position du curseur de la source de donnée (BDS) et l'enregistre en base de donnée
  - Initialisation du mode de checkpoint au lancement

- 3 algorithmes
  - Sur le temps
  - Sur le nombre d'enregistrement
  - Par programmation

Un modèle de programmation Batch simplifié …

Archive et Déploiement

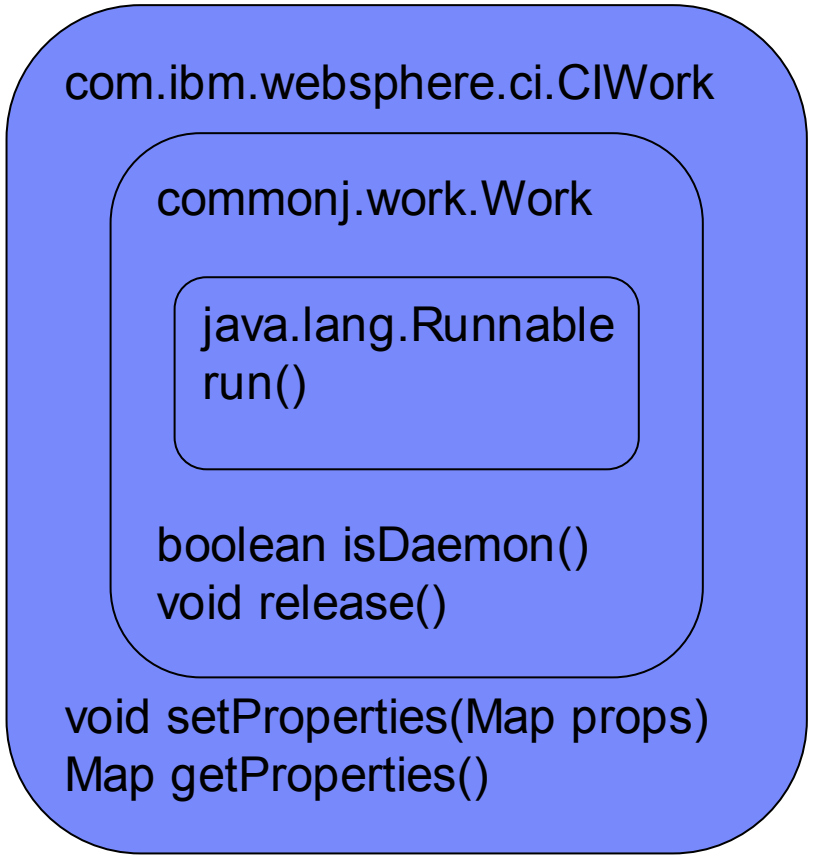


Agenda

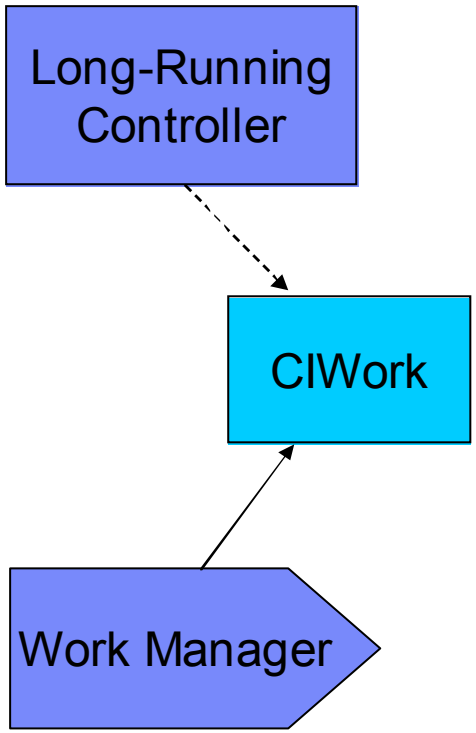
- Introduction
- 3 types de traitement
  - Batch Transactionnel
  - Cacul intensif
  - Natif
- Environment

Modèle de programmation calcul intensif

- Pour chaque étape du Batch on indique le nom de la classe qui implémente l'interface com.ibm.websphere.ci.CIWork
  - Héritage de commonj.work.Work
- Spécificités de CIWork:
  - La methode isDaemon() avec un return true.
  - Pas d'argument sur le constructeur
  - Work.release()



Compute-intensive execution environment



- Chaque traitement (step) du job, dans un EJB Stateless, avec les étapes suivantes :
  - Instanciation d'un CIWork assynch bean
    - Exécution du constructeur sans argument
  - Récupération des paramètres du xJCL
    - Exécution de la méthode setProperties(Map)
  - Exécution des traitements dans des pools auto-géré
    - Exécution de la méthode Run()
    - Exécution de la méthode Release() après annulation

    Pour pouvoir si besoin ré-exécuter la méthode run()

Agenda

- Introduction
- 3 types de traitement
  - Batch Transactionnel
  - Calcul intensif
  - Natif
- Environment

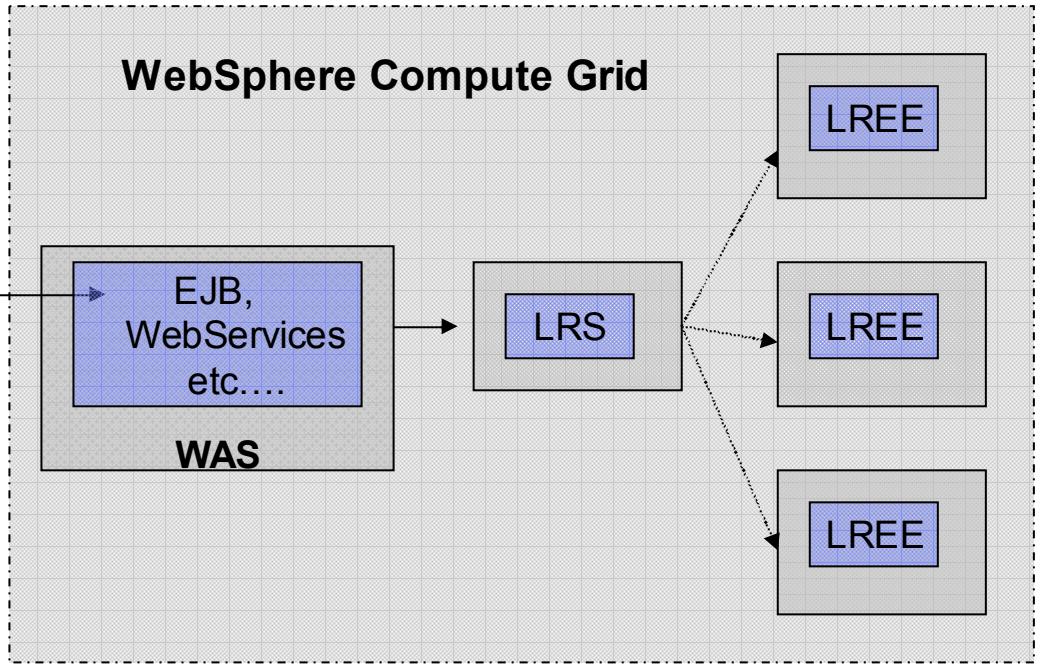
Application batch native

- Applications
  - Pas de contrainte de langage, d'API
  - Déployée séparément
  - Son propre environnement d'exécution
  - Variables
    - Configuration par défaut de la platforme
    - custom properties du noeud 'grid.apps' & 'grid.env'
    - Définition dans le xJCL
- Nœuds génériques WebSphere

Pourquoi “Grid”??

Invocation de la méthode exposée, par exemple :

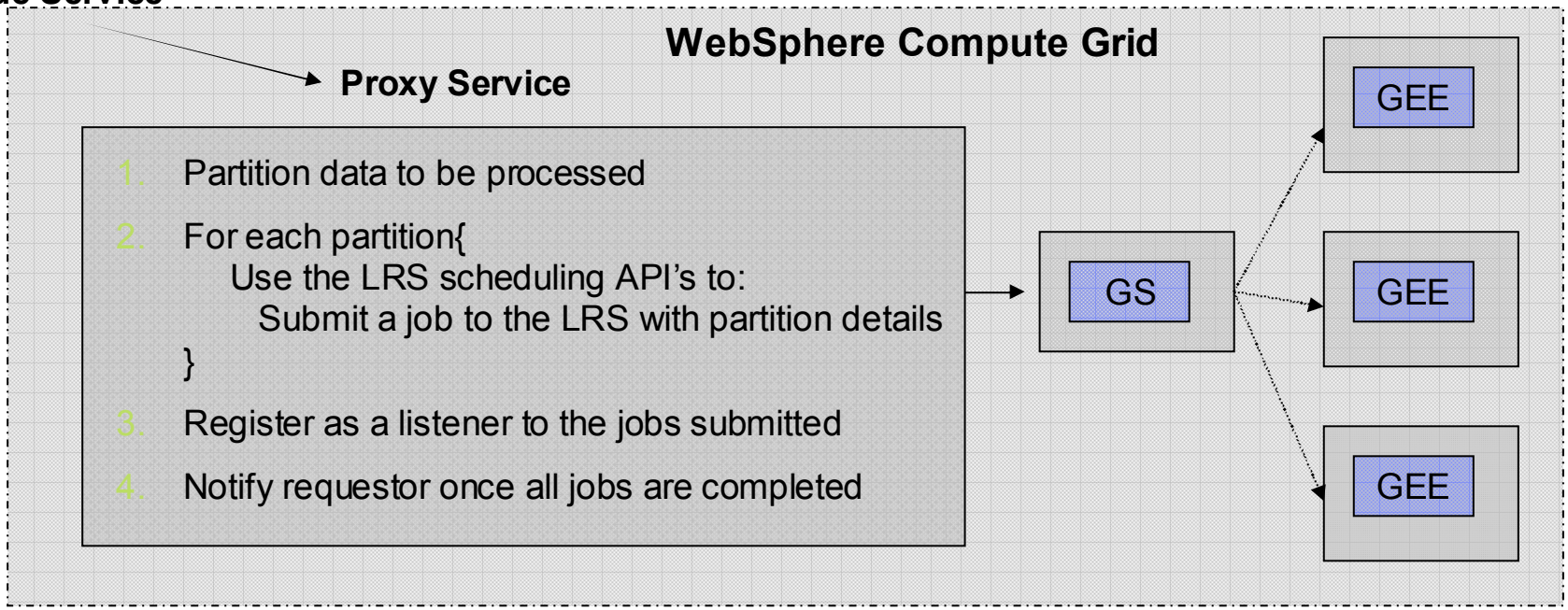
ExecuteEndOfMonthProcessing()



- Vaut mieux exposer pour la réutilisation une méthode d’un POJO, EJB, Jms, WS que les appels natifs du framewok (LRS, xJCL, et infra WCG Batch infrastructure)
- Les autres applications peuvent initier les batchs sans aucun développement spécifiques

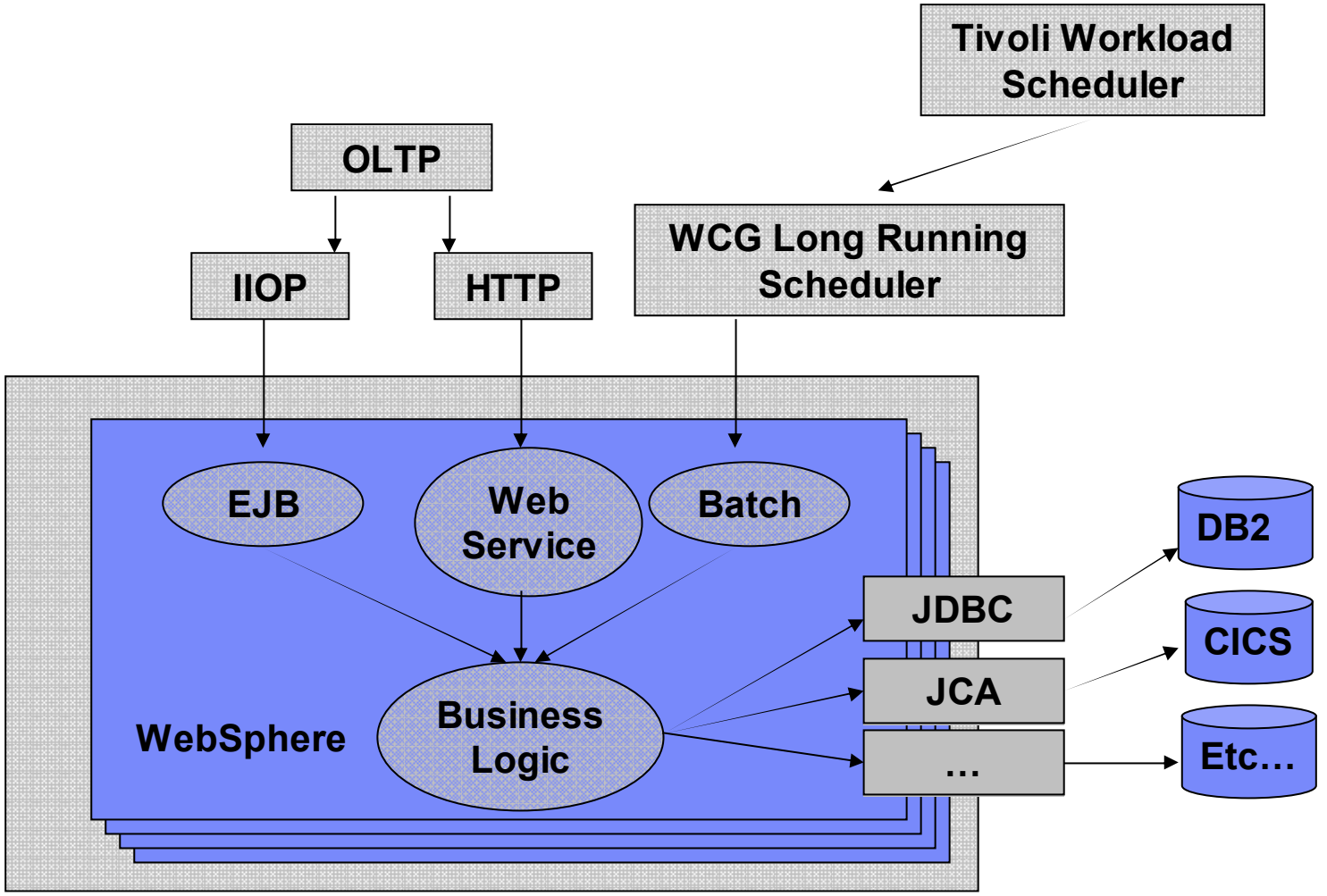
Pourquoi “grid” ??

Invocation d’un Proxy de Service



- Le proxy peut subdivisé la requête en plusieurs traitements
- Chaque demande de traitement est envoyée à l’ordonnanceur
- L’ordonnanceur (GS) **parallélisent** les traitements sur les moteurs d ’exécution (GEE).
- L’état de chaque traitement est renvoyé sur le proxy de service

Coexistence application OLTP et Batch



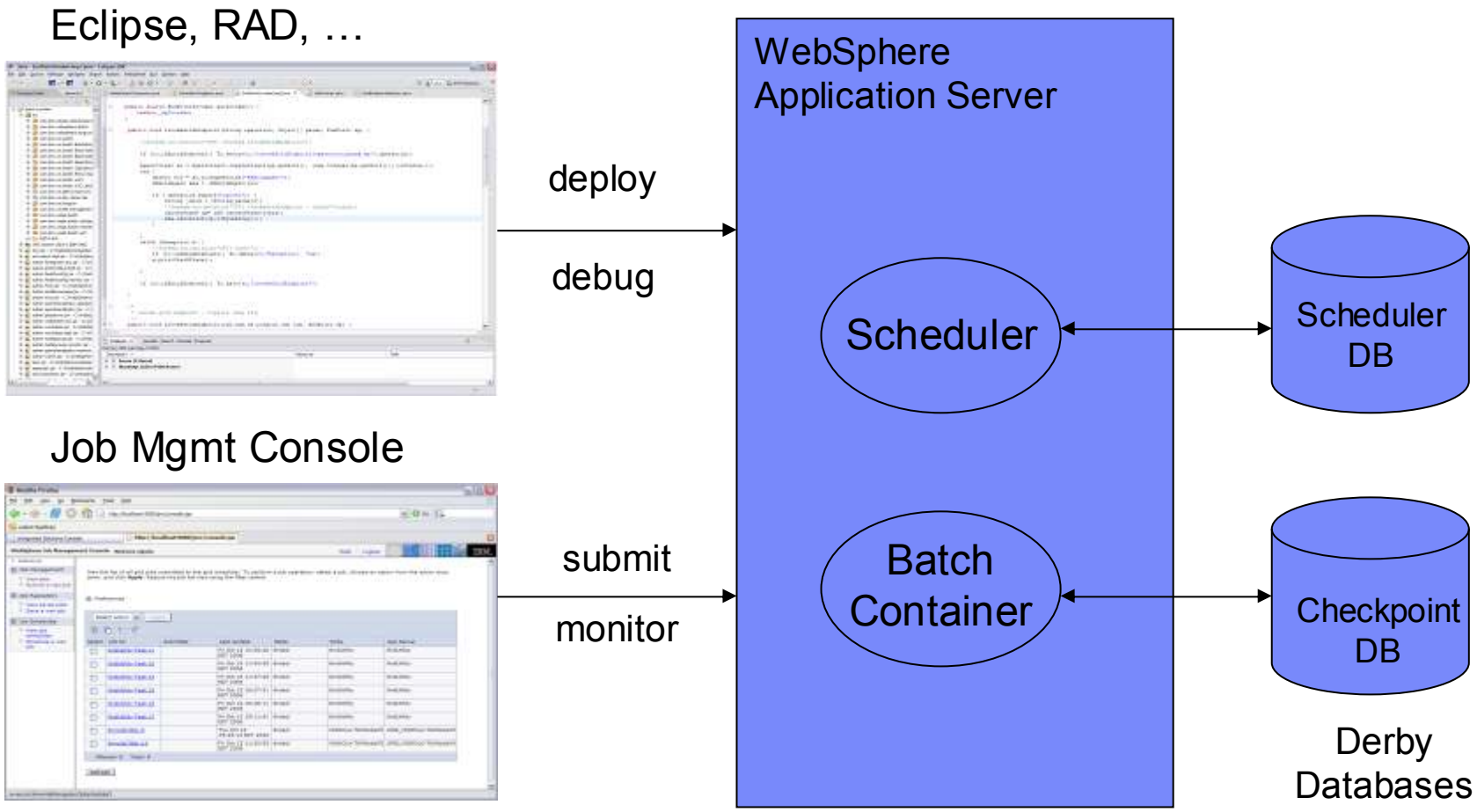
Agenda

- Introduction
- 3 types de traitement
  - Batch Transactionnel
  - Calcul intensif
  - Natif
- Environnement

Installation

- WebSphere
  - les applications Grids sont déployées comme des applications JEE
  - Lorsque l'application est déployée, WebSphere CG reconnait automatiquement que c'est une application Grid
  - Le process d'installation va répliquer les binaires de l'applications sur les environnements d'exécution
  - Les applications Web et Grid peuvent cohabiter sur les mêmes serveur d'application
  - WVE peut aussi bien gérer dynamiquement l'affectation des application Web et Grid sur un pool commun de noeud
- Possiblité de définir des qualités de service

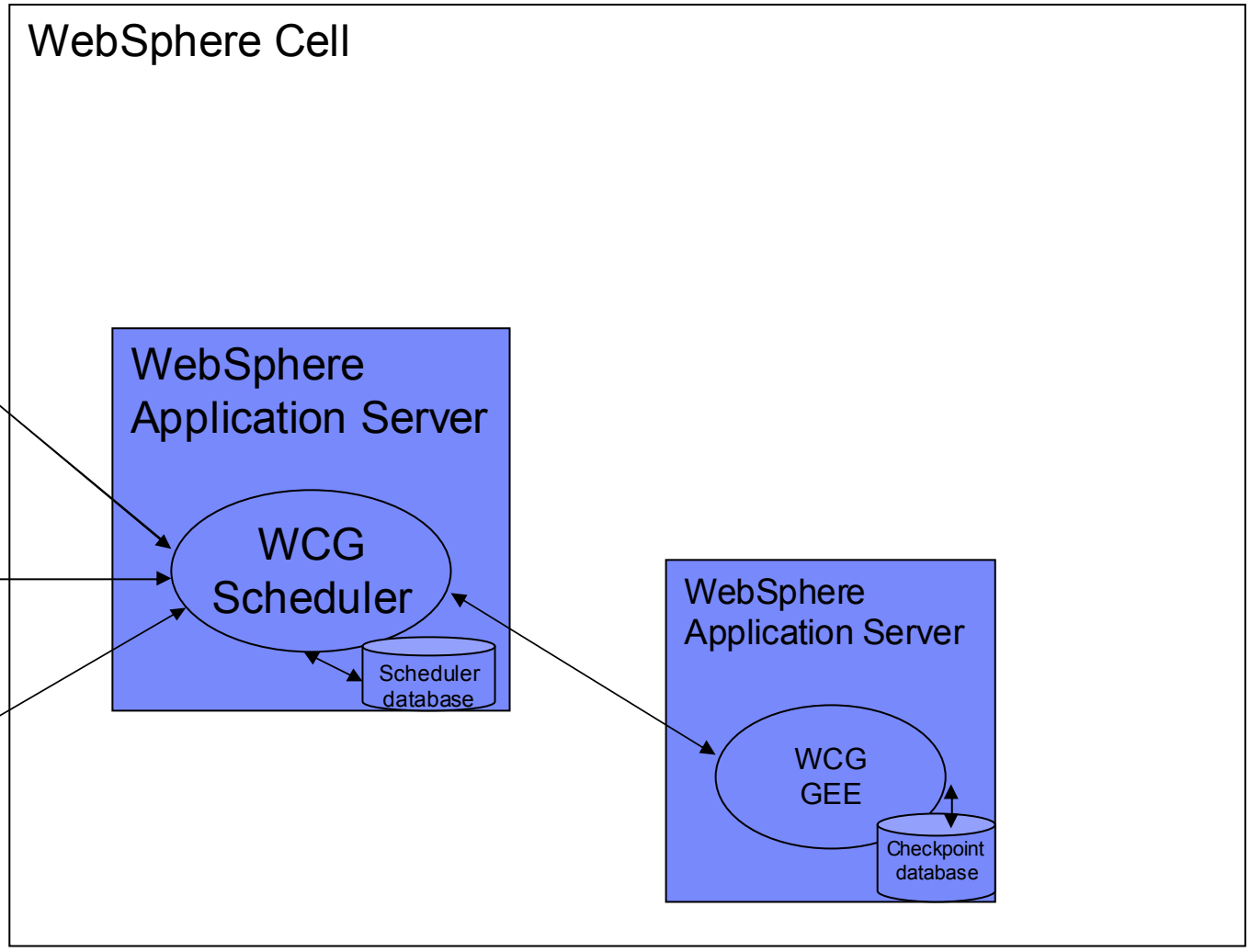
Unit Test Environment



WebSphere WCG – Topologies

- console
- command line
- APIs

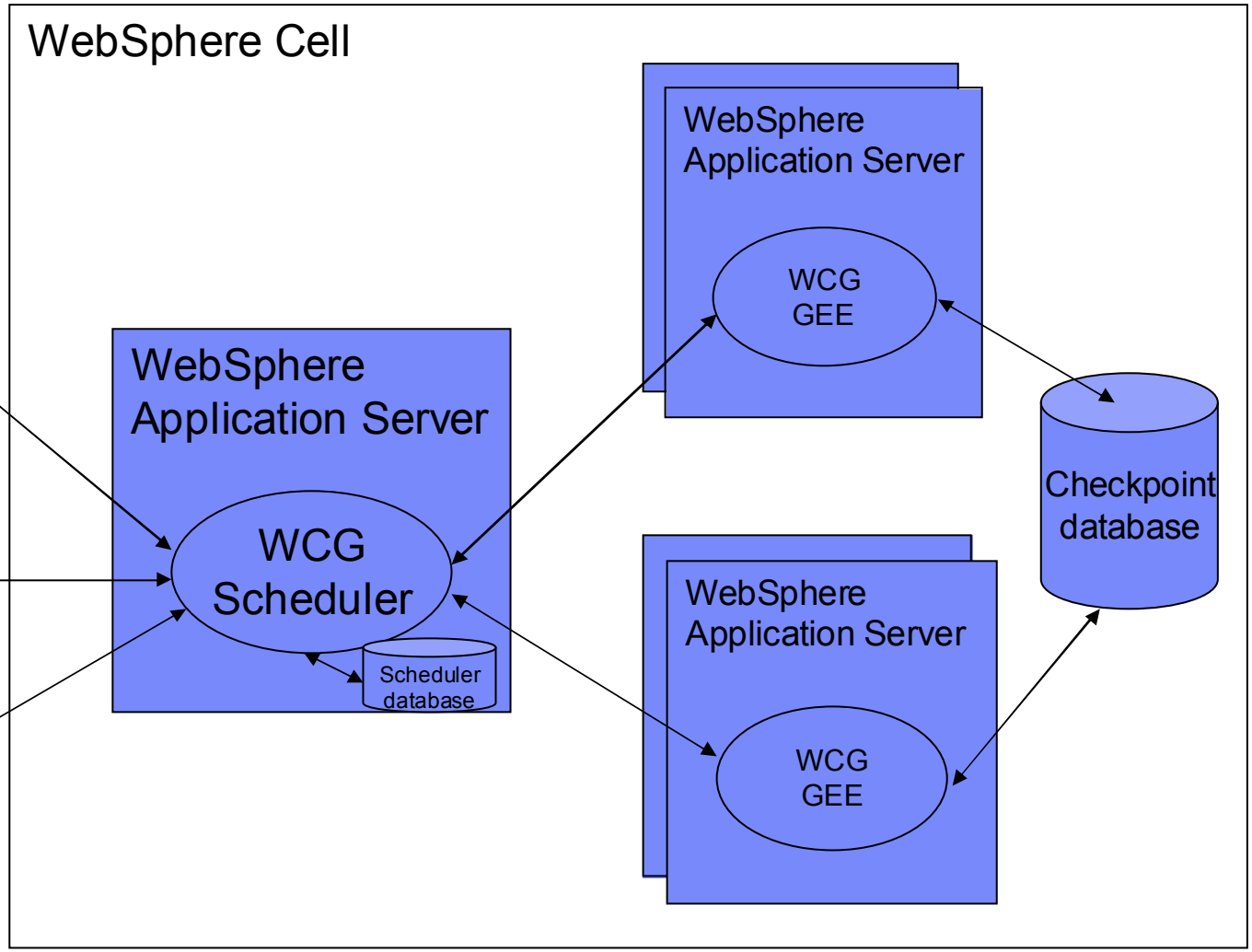
```
public getJobLog(String jobid) {
   _scheduler.getJobLog(jobid);
}
```



WebSphere WCG – Topologies

- console
- command line
- APIs

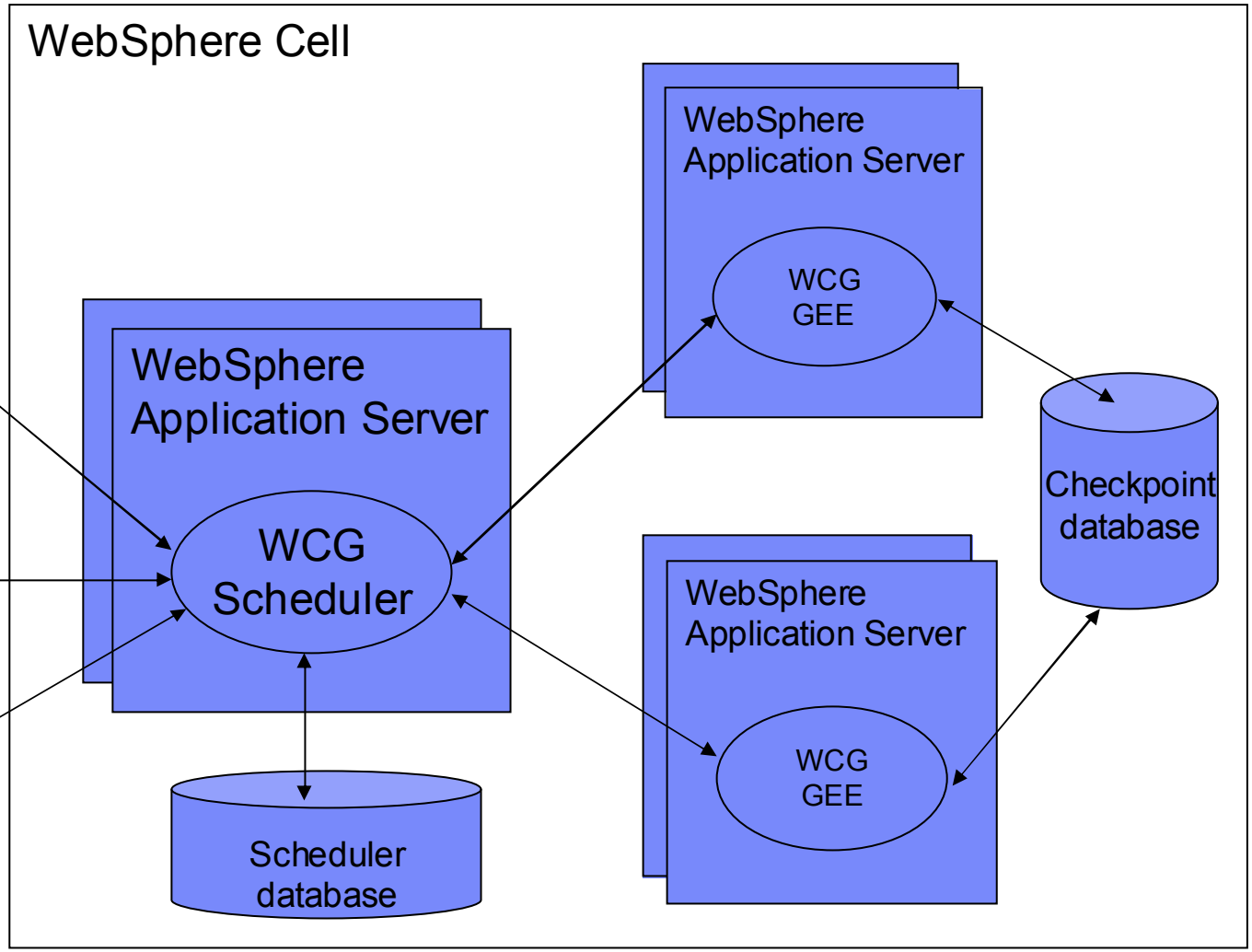
```
public getJobLog(String jobid) {
   _scheduler.getJobLog(jobid);
}
```



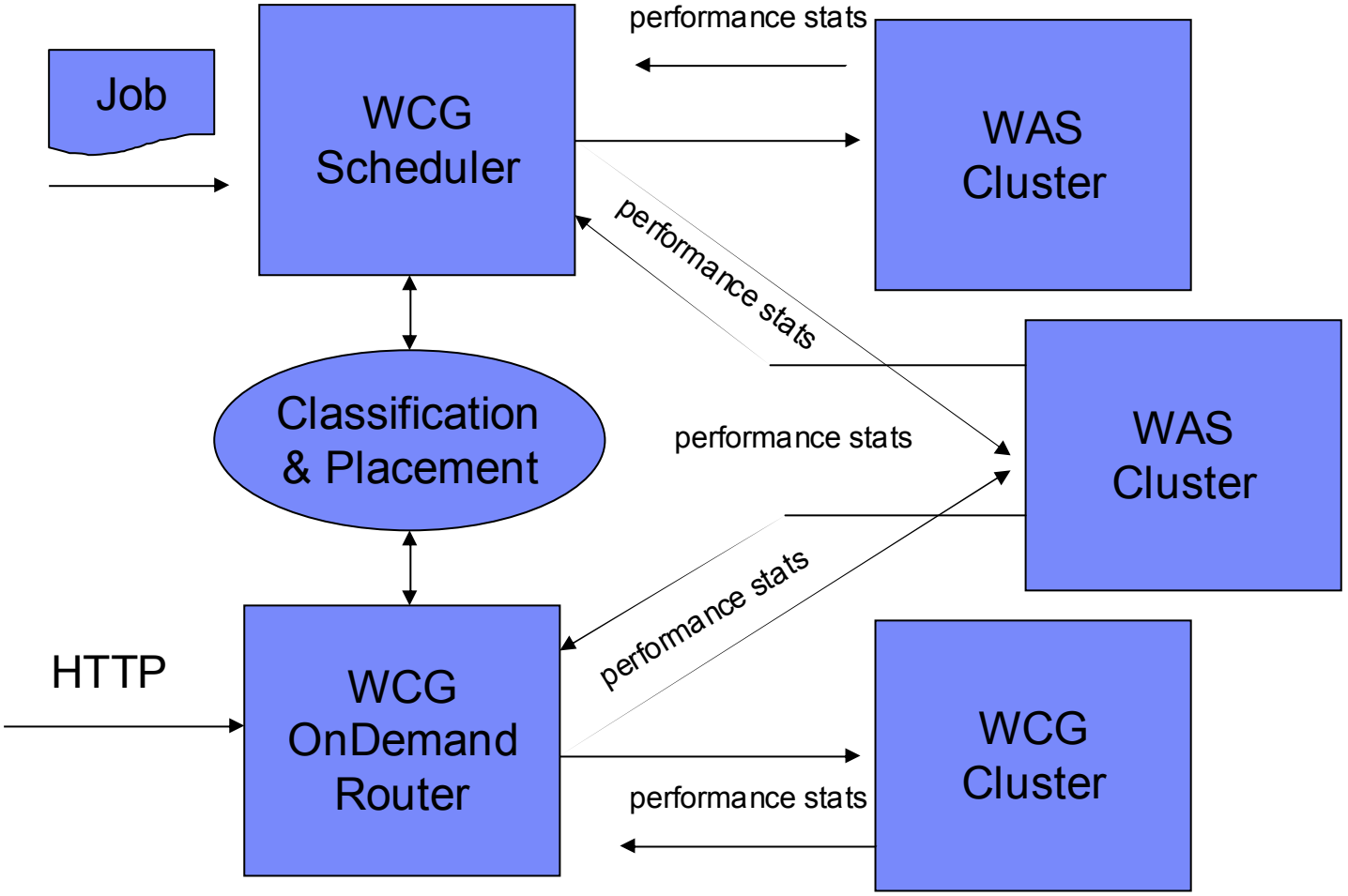
WebSphere WCG – Topologies

- console
- command line
- APIs

```
public getJobLog(String jobid) {
   _scheduler.getJobLog(jobid);
}
```



Classification des requêtes et affectation des traitements



Interfaces de l'ordonnanceur

- Peut être invoqué par :
  - La console d'administration
  - Le mode lige de commande
  - Par programmation en EJB/WS
- Pour administrer et superviser
  - Toutes les opérations sont disponibles via chaque interface

Console d'administration WCG

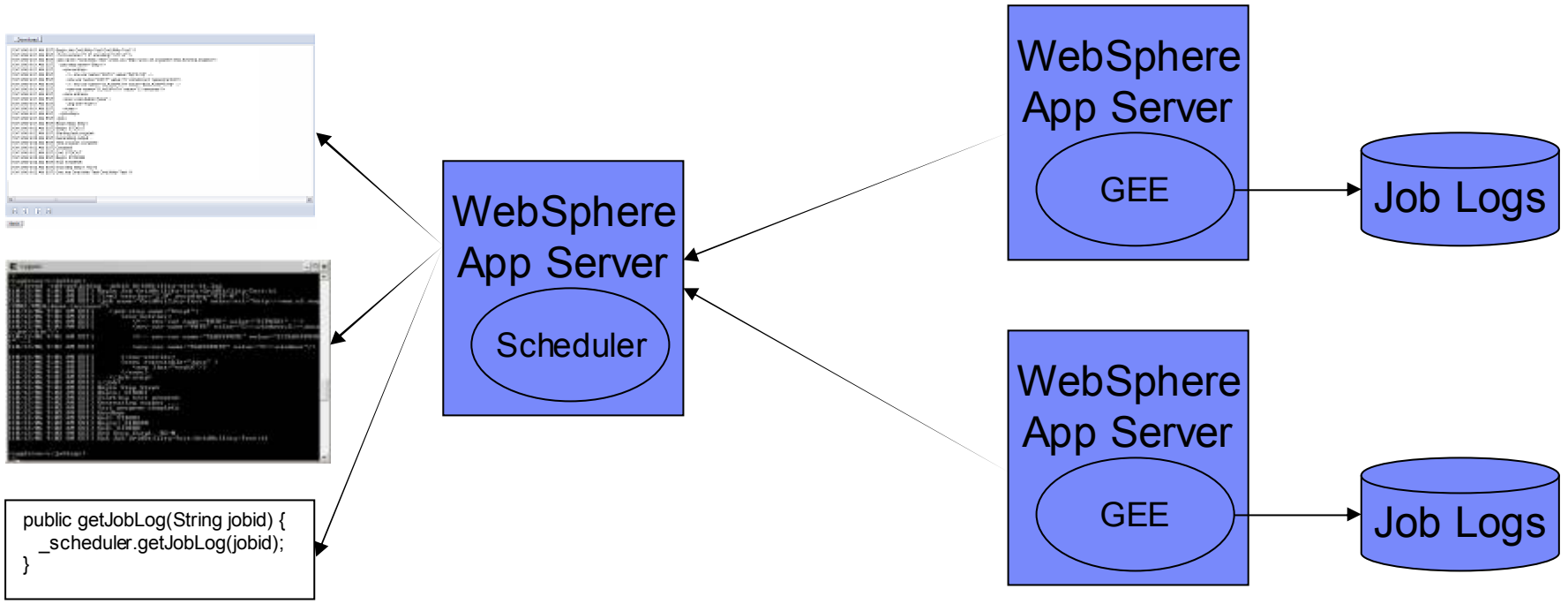
- Une interface Web en frontal de l'ordonnanceur
  - Sur le même serveur que l'ordonnanceur
- Permet :
  - La soumission des traitements
  - Les opérations
    - cancel, stop, suspend, resume, restart, purge
  - La gestion du registre de définition
    - save, delete
  - Programmer le lancement de traitement
    - create, delete job schedules
- Securité
  - User/password
  - Rôle lrsubmitter, lrAdmin

Rôles Compute Grid

Action	Admin	Submitter
Submit xJCL as Job	Yes	Yes
Submit Job by name	Yes	Yes
Cancel Job	Yes	Only owned jobs
Purge Job	Yes	Only owned jobs
Download Job xJCL	Yes	Only owned jobs
Restart	Yes	Only owned jobs
View Job Status (All)	Yes	Only owned jobs
Save named Job xJCL	Yes	
Remove named job xJCL	Yes	
View xJCL for named Job	Yes	Yes

Les Logs

- Sur les systèmes de fichier des moteurs
- Peuvent être accédées en remote via la console, ligne de commande ou les APIs de l'ordonnanceur



Classifications (WCG + WVE)

- Contrôle des ressources consommées
- Les critères
  - Durée maximale d'exécution
  - Nombre maximum de traitement sur un moteur d'exécution
  - Taille maximale des fichiers de logs
  - Rétention de logs (âge, espace)
  - Rétention maximale d'un enregistrement d'exécution (âge, nombre)
- Assigné via des propriétés dans le xJCL
- Peut être surclassé par programmation

Règles de classification (WCG + WVE)

- Approche SLA
- Définies dans la console d'administration WAS au niveau des propriétés de l'ordonnanceur
- dans un scope cellule
- Evaluées dans un ordre défini
- Objectif premier est d'atteindre les politiques de services
- Règles de type booléenne en utilisant les opérateurs et les arguments :
  - Nom du job
  - Class du job
  - Utilisateur et groupe
  - heure, date
  - Plate-forme (e.g. z/OS)

Refacturation

- 2 options
  - Enregistré dans la base de donné de l'ordonnanceur. Via l'interface MBean possibilité d'exporter un fichier format CVS (importable vers Tivoli ITUAM)
  - type SMF Record 120
  - Les deux choix sont disponibles sur z/OS
- Statistiques collectées par job
  - Nom du job
  - id
  - heure/Date (start/end)
  - Emetteur
  - cell/noeud/serveur
  - Utilisation CPU

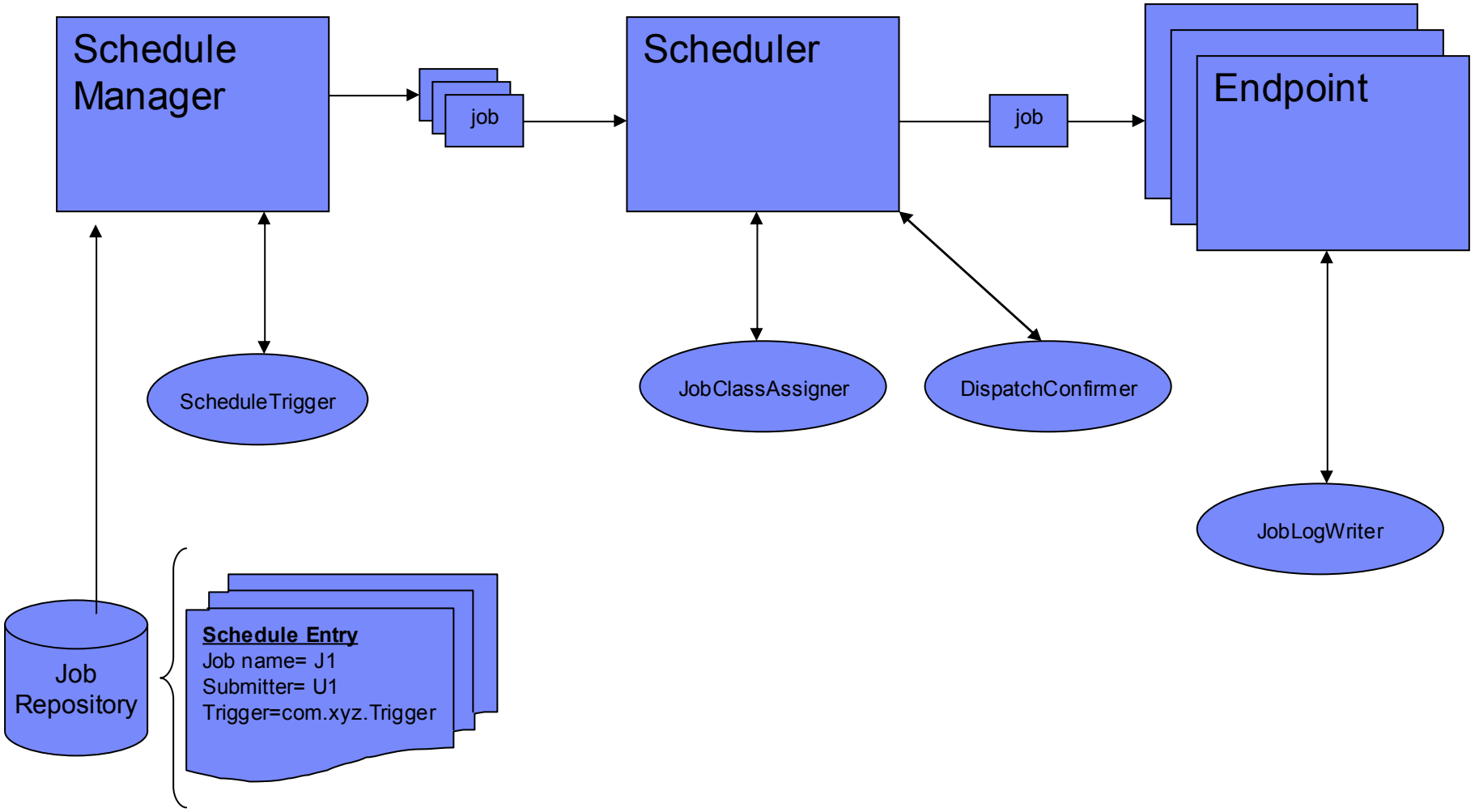
Intégration vers un ordonnanceur d'entreprise

- WSGrid
  - Lancement et remontée des bacths via des ordonnanceurs extérieurs (TWS, Control-M, etc)
  - Inclure les batchs WCG dans des séquences d'action plus génériques
  - L'ordonnanceur WCG devient un proxy de service
  - Une interface en ligne de commande
  - Plus besoin du fichier xJCl

Business Grid SPIs

- ScheduleTrigger
  - custom rule for submitting a scheduled job
  - allows for submission conditions beyond time/date
- JobClassAssigner
  - custom logic for validating job class usage
  - allows for override of user-specified job class
- DispatchConfirmer
  - custom logic for confirming XD dispatch decision
  - multiple actions possible:
    - cancel job
    - re-queue job for later dispatch
    - specify target endpoint
- JobLogInterceptor
  - allows for modification of job log content
    - Edit
    - Delete
    - System log vs job log only
  - Can replace file-based logs with alternative destination
- JobNotificationListener
  - Receives notifications for key job lifecycle events:
  - job start/end
  - step start/end

Business Grid SPIs …



Questions