ZOS: Travaux par lots Java sur z/OS et OS/390

Table of Contents

- ZOS: Travaux par lots Java sur z/OS et OS/390
  - Restrictions relatives aux travaux par lots Java
    - Configuration flexible de l'environnement
    - Routage de la sortie
    - Encodage de la sortie de contrôle
    - Utilisation de jeux de données MVS et d'instructions DD
    - Exécuter la machine virtuelle Java dans le même espace d'adressage
    - Communication avec le journal de travail MVS et la console système
  - Utilitaire BPXBATCH
    - Exécution de travaux par lots Java avec BPXBATCH
    - Variables d'environnement utilisées par Java
  - Comparaison des outils de lancement de travaux Java par lots

Java est présent sur le mainframe sous de nombreuses formes - le plus visible dans WebSphere - mais joue également un rôle important pour des produits tels que CICS, DB2 et IMS. Jusqu'à récemment, toutefois, Java n'était pas très visible dans l'environnement de traitement par lots, sans doute le type de charge de travail le plus répandu de z/OS. Avec l'annonce récente de zAPP, attendez-vous à ce que cela change.

Dans cet article, nous décrivons certaines des difficultés que vous pouvez rencontrer lors de l'utilisation de Java dans des flux de travaux par lots et comment intégrer de manière transparente Java dans le langage JCL (MVC).

Restrictions relatives aux travaux par lots Java

Configuration flexible de l'environnement

Comme pour les autres environnements Java, les SDK Java pour z/OS exigent que les répertoires et les fichiers JAR contenant les classes soient répertoriés dans la variable d'environnement CLASSPATH. Ces fichiers doivent obligatoirement se trouver dans le système de fichiers hiérarchique (HFS) ou dans le système de fichiers zSeries (zSeries).

Routage de la sortie

Le routage de la sortie de votre travail par lots aux ensembles de données MVS normaux, y compris les ensembles de données JES SYSOUT (pour Java, cela inclurait la sortie `System.out` et `System.err`) n'est possible qu'avec le lanceur JVM personnalisé (JZOS)

Encodage de la sortie de contrôle

L'encodage des caractères de la sortie dutravail doit être contrôlé, car la machine virtuelle Java s'exécute souvent avec ASCII comme encodage par défaut:

- en interne, Java s’exécute toujours avec des caractères codés en Unicode, mais la propriété système `file.encoding` spécifie le jeu de caractères utilisé lors de la conversion de caractères Java de et vers. octets pour l'entrée et la sortie.
- sous z/OS, la variable d'environnement LANG détermine le codage par défaut, qui est normalement IBM-1047 (EBCDIC).

Certaines applications Java, cependant, supposent à tort que le codage par défaut est une page de code ASCII telle que ISO-8859-1. Pour que ces applications fonctionnent correctement, il faut définir la propriété `file.encoding` sur ISO-8859-1. S'il n'y avait aucun moyen de contrôler séparément le codage d'une sortie système Java, la sortie système serait également écrite en ASCII. Par conséquent, on a besoin d'un moyen de contrôler le codage de la sortie d'un travail séparément du codage JVM par défaut.

Utilisation de jeux de données MVS et d'instructions DD

Le package standard java.io ne fonctionne qu'avec les fichiers HFS ou zFS, mais il serait intéressant de pouvoir utiliser les jeux de données MVS traditionnels, soit par nom, soit en utilisant des cartes DD dans le JCL. Cette fonctionnalité n'est utilisable qu'avec le lanceur JVM personnalisé (JZOS)

Exécuter la machine virtuelle Java dans le même espace d'adressage

Chaque nouveau processus UNIX peut créer un espace adresse OMVS distinct. Ce comportement pose souvent des problèmes dans le mode des lots MVS traditionnel. Tout d'abord, les espaces adresse ne partagent pas les noms DD, les fichiers SYSOUT et les autres allocations de jeu de données. Si la machine virtuelle Java est lancée dans un espace d'adressage séparé, les objectifs présentés ici peuvent devenir difficiles à atteindre. En outre, la surveillance des travaux et la gestion des performances peuvent être délicates si plusieurs espaces adresse sont utilisés. Des données de comptabilité, telles que des enregistrements de type 30 du cadre de gestion de service (SMF), seront créées pour chaque espace adresse, ce qui est souvent indésirable.

Communication avec le journal de travail MVS et la console système

Il est courant que les travaux par lots utilisent la macro Écriture dans un opérateur (WTO) pour écrire des messages dans le journal des travaux et la console système. Pour les travaux de longue durée et les tâches démarrées, l’opérateur système peut souhaiter arrêter le programme en utilisant la commande `MVS STOP (P)` ou lui envoyer des commandes arbitraires à l’aide de la commande `MODIFY (F)`.

Utilitaire BPXBATCH

On peut utiliser l'utilitaire **BPXBATCH**, qui fait partie des services système Unix, pour lancer n'importe quelle commande de shell UNIX, y compris la commande de shell Java faisant partie du SDK. Le listing 1 ci-dessous illustre un travail utilisant **BPXBATCH** pour démarrer un programme Java et envoyer sa sortie aux jeux de données JES SYSOUT:

Exécution de travaux par lots Java avec BPXBATCH

```
//BPXBATCH JOB (999,XXX),'JAVA BPXBATCH',CLASS=A,MSGLEVEL=(1,1)
//   MSGCLASS=X,REGION=0M,NOTIFY=&SYSUID
//*********************************************************************
//* Run Java under a UNIX System Service shell
//*********************************************************************
//STEP2 EXEC PGM=BPXBATCH,
// PARM='SH java com.foo.MyClass arg1 arg2'
//STDIN  DD DUMMY
//STDOUT DD PATH='/tmp/&SYSUID..bpxbatch.out',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDERR DD PATH='/tmp/&SYSUID..bpxbatch.err',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDENV DD *
CLASSPATH=/u/myuid/classes
//*********************************************************************
//* Copy HFS output files to SYSOUT, since BPXBATCH can only write
//* STDOUT and STDERR to HFS files.
//*********************************************************************
//STEP3 EXEC PGM=IKJEFT01,DYNAMNBR=300,COND=EVEN
//SYSTSPRT DD SYSOUT=*
//HFSOUT DD PATH='/tmp/&SYSUID..bpxbatch.out'
//HFSERR DD PATH='/tmp/&SYSUID..bpxbatcherr'
//STDOUTL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//STDERRL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
OCOPY INDD(HFSOUT) OUTDD(STDOUTL)
OCOPY INDD(HFSERR) OUTDD(STDERRL)
//
```

- **Configuration flexible de l'environnement**: des variables d'environnement, telles que CLASSPATH, peuvent être spécifiées dans le script de connexion par défaut `.profile` de l'utilisateur ou à l'aide de `// STDENV DD`. Le jeu de données `// STDENV`, cependant, ne permet aucune substitution de variable ni aucun script.
- **Routage de sortie**: les flux Java `System.out` et `System.err` sont envoyés à `// STDOUT` et `// STDERR`. **BPXBATCH** ne prend en charge que les fichiers HFS pour ces DD. Par conséquent, une étape distincte est requise pour copier ces fichiers dans des ensembles de données

SYSOUT. Cela signifie également que vous ne pouvez pas afficher la sortie tant que le travail n'est pas terminé.
- **Contrôle de l'encodage de la sortie**: l'encodage de la sortie par défaut ne pouvant pas être contrôlé séparément de la propriété système `file.encoding par défaut`, une étape distincte est nécessaire pour effectuer la conversion.
- **Utilisation de jeux de données MVS et d'instructions DD**: on ne peut pas utiliser desjeus de données puremnt MVS.
- **Passage du code de condition**: on ne peut pas transmettre de code de condition Java à l'aide de `System.exit`) aux étapes suivantes.
- **Exécuter la machine virtuelle Java dans le même espace adresse**: Par défaut, BPXBATCH démarre la machine virtuelle Java dans un espace adresse distinct, afin que le programme Java n'ait pas accès aux allocations de jeu de données DD dans l'étape de travail.

On peut utiliser une version spéciale (point d'entrée) de l'utilitaire BPXBATCH, BPXBATSL, pour que le shell se crée dans le même espace d'adressage afin de résoudre ce problème. BPXBATSL a une limitation qui l'empêche d'exécuter un shell d'ouverture de session s'il n'est pas exécuté en tant que root. Il doit donc être exécuté comme suit:

```
//STEP2 EXEC PGM=BPXBATSL,
// PARM='PGM /bin/sh -c java com.foo.MyClass arg1 arg2'
```

Ceci nécessite également que `//  STDENV` inclue des paramètres pour toutes les variables d'environnement, car `/etc/profile` et `.profile` ne seront pas exécutés. Pour cette raison, il est souvent préférable d'exécuter un script shell pour gérer cela plutôt que d'appeler directement Java.

- **Communication avec le journal des travaux et la console système**: BPXBATCH ou le SDK Java ne disposent d'aucune possibilité d'écrire des messages WTO ou de répondre aux commandes de l'opérateur STOP ou MODIFY. Un programme Java peut faire un appel JNI à un programme C qui peut ensuite appeler les fonctions C _console ou _console2 pour émettre des WTO ou interagir avec STOP et MODIFY. Voir Communication avec la console système MVS pour plus d'informations.

Variables d'environnement utilisées par Java

Les variables d'environnement couramment utilisées dans les applications Java sont les suivantes:

- **PATH**: Liste des répertoires, séparés par deux points (“:”), utilisés pour rechercher les fichiers binaires exécutables. Cela devrait inclure $JAVA_HOME/bin.
- **LIBPATH**: Liste de répertoires, séparés par deux points, utilisés pour rechercher des bibliothèques partagées. Cela devrait inclure toutes les bibliothèques requises par les bibliothèques JNI utilisées par votre application. Lors de l'exécution du programme de lancement JZOS, cela doit inclure:

```
$JAVA_HOME/bin:$JAVA_HOME/bin/classic:$JZOS_HOME
```

- **CLASSPATH JZOS Batch Launcher**: Liste des répertoires et des fichiers JAR/ZIP, séparés par un signe deux-points, à partir desquels charger les classes Java. Cela devrait inclure les répertoires ou les fichiers JAR individuels utilisés par votre application Java. Lors de l'exécution du programme de lancement par lots JZOS, ceci doit également inclure $JZOS_HOME/jzos.jar.
- **IBM_JAVA_OPTIONS**: Liste d'options de la machine virtuelle Java séparées par des espaces. Tapez Java -help à partir d'un z/OS ou OMVS shell pour afficher une liste d'options. Par exemple:

```
IBM_JAVA_OPTIONS="-Xms64m -Xmx128m -Djzos.home=/u/jzos"
```

Comparaison des outils de lancement de travaux Java par lots

	BPXBATCH	**BPXBATSL**	**Lanceur JVM personnalisé (JZOS)**
Configuration flexible des variables d'environnement	Oui, la substitution de variable et les scripts ne sont pas autorisés	Oui, la substitution de variables et les scripts ne sont pas autorisés	Oui
Routage du répertoire de sortie vers les jeux de données SYSOUT	Non	Non	Oui
Contrôle du codage de la sortie séparément du codage JVM par défaut	Oui, en utilisant iconv	Oui, en utilisant iconv	Oui
Code de condition passant entre les étapes Java et non Java	Non	Non	Oui
Utiliser des jeux de données MVS et des Déclarations DD	Non	Oui	Oui
Exécuter La machine virtuelle Java dans le même espace adresse	Non	Oui	Oui
Communication avec la console MVS	Non	Oui, utilisation de la fonction `\_console` à partir d'une routine JNI	Oui