VISHNU - Modification pour l'ajout d'un nouveau batch scheduler dans TMS

COLLABORATORS

	TITLE : VISHNU - Modification pour l'ajout d'un nouveau batch scheduler dans TMS	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Daouda Traoré	April 4, 2012	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
1	29/03/2012	Version initiale	D. TRAORE

Contents

Chapter 1

Ajout d'un nouveau batch scheduler dans TMS

1.1 Objectifs du document

L'objectif de ce document est de présenter les motifications à apporter dans TMS pour l'ajout d'un nouveau batch scheduler.

1.2 Structure du document

Ce document contient les parties suivantes:

- Construction des classes concrètes
- Construction de l'analyseur de script (parser)
- Modification de la classe fabrique
- Modification des fichiers permettant de construire des exécutables (tmssed et tmslave)
- Modification de la classe SSHJobExec
- Modifications des CMakeFiles
- Résumé

Chapter 2

Construction des classes concrètes

2.1 La classe abstraite BatchServer

La classe implémentant le nouveau batch à ajouter dans TMS doit hériter de la classe abstraite **BatchServer** définie dans *TMS/src/server* . Cette classe abstraite déclare les méthodes virtuelles nécessaires pour la gestion des tâches dans TMS. Le document d'achitecture générale de TMS (*TMS/design/docbook/TmsDesign-gen.docbook.pdf*) donne plus de détails sur cette classe. Ci-dessous un exemple pour LSF :

```
#ifndef TMS_LSF_SERVER_H
#define TMS_LSF_SERVER_H
  #include <lsf/lsbatch.h>
  #include "BatchServer.hpp"
  /**
  * \class LSFServer
  * \brief LSFServer class implementation
  */
  class LSFServer : public BatchServer
  {
  public:

    LSFServer();

    int submit(const char* scriptPath,

    const TMS_Data::SubmitOptions& options,TMS_Data::Job& job,char** envp=NULL);

    int cancel(const char* jobId) ;

    int getJobState(const std::string& jobId);

    time_t getJobStartTime(const std::string& jobId);

    TMS_Data::ListQueues* listQueues(const std::string& optQueueName=std::string ↩
        ());

    void fillListOfJobs(TMS_Data::ListJobs*& listOfJobs,
    const std::vector<string>& ignoredIds=std::vector<string>());

    ~LSFServer();
  }
#endif
```

Chapter 3

Construction de l'analyseur de script (parser)

La directive permettant de récupérer les options et leurs valeurs associées dans le script à soumettre est spécifique à chaque batch scheduler. Par exemple la directive **LSF** est *#BSUB*. Pour pouvoir récupérer ces options et leurs valeurs, le script à soumettre doit être analysé. Certains batchs (LoadLeveler par exemple) analysent le script dans leur fonction de soumission, et dans ce cas le developpeur ne fera qu'appeler l'api de la fonction de soumission avec le script. Pour les autres batchs schedulers ne permettant pas l'analyse à travers la fonction de soumission le devéloppeur doit obligatoirement analyser le script et remplir la structure à fournir à la fonction de soumission avec les options et leurs valeurs associées récupérées dans le script. Le devéloppeur pourra se référer aux analyseurs proposés dans le code source des batchs qui sont des logiciels libres et écrire lui même l'analyseur (*parser*) si le batch est un logicel commercial (exemple dans **TMS/src/lsf_parser**).

Chapter 4

Modification de la classe fabrique

La seule modification à ajouter dans cette classe est l'ajout dans le fichier **TMS/src/server/BatchFactory.cpp** le cas de test correspondant à la création de l'instance du type du nouveau batch dans la méthode ***getBatchServerInstance()***. Le nouveau type de batch doit aussi être ajouté dans le type ***enum BatchType*** défini dans le **core/src/utils/utilVishnu.hpp** (faire attention aux numéros des types enum pendant l'ajout). Ci-dessous les créations des instances de quatres batchs (TORQUE, LOADLEVELER, SLURM et LSF).

```
BatchServer*
BatchFactory::getBatchServerInstance(BatchType batchType) {
  switch (batchType){
    case TORQUE:
    #ifdef HAVE_TORQUE
    mbatchServer = new TorqueServer();
    #else
    mbatchServer = NULL;
    #endif
    break;
    case LOADLEVELER:
    #ifdef HAVE_LOADLEVELER
    mbatchServer = new LLServer();
    #else
    mbatchServer = NULL;
    #endif
    break;
    case SLURM:
    #ifdef HAVE_SLURM
    mbatchServer = new SlurmServer();
    #else
    mbatchServer = NULL;
    #endif
    break;
    case LSF:
    #ifdef HAVE_LSF
    mbatchServer = new LSFServer();
    #else
    mbatchServer = NULL;
    #endif
    break;
    default:
    mbatchServer = NULL;
    break;
  }
  return mbatchServer;
}
```

enum BatchType défini dans le **core/src/utils/utilVishnu.hpp** dans l'état actuel :

```
/**
* \enum BatchType
* \brief The type of the Batch
*/
typedef enum {
TORQUE = 0,/*!< For TORQUE batch type */
LOADLEVELER = 1,/*!< For LOADLEVELER batch type */
SLURM = 2, /*!< For SLURM batch type */
LSF = 3, /*!< For LSF batch type */
UNDEFINED = 4 /*!< IF batch type is not defined*/
} BatchType;
```

Chapter 5

Modification des fichiers permettant de construire des exécutables (tmssed et tmslave)

La modification à ajouter dans le fichier **TMS/src/sed/tmssed.cpp** est le cas de test permettant d'initialiser le type du nouveau batch. Ci-dessous un exemple montrant les initialisations de quatres batchs (TORQUE, LOADLEVELER, SLUM et LSF).

```
        #ifndef HAVE_TORQUE
        std::cerr << std::endl;
        std::cerr << "Error: can't initialize TORQUE batch type because this server has not ↩
             compiled with TORQUE library" << std::endl;
        std::cerr << std::endl;
        exit(1);
        #endif
        batchType = TORQUE;
        } else if (batchTypeStr == "LOADLEVELER") {
        #ifndef HAVE_LOADLEVELER
        std::cerr << std::endl;
        std::cerr << "Error: can't initialize LOADLEVELER batch type because this server  ↩
            has not compiled with LOADLEVELER library"  << std::endl;
        std::cerr << std::endl;
        exit(1);
        #endif
        batchType = LOADLEVELER;
        } else if (batchTypeStr == "SLURM") {
        #ifndef HAVE_SLURM
        std::cerr << std::endl;
        std::cerr << "Error: can't initialize SLURM batch type because this server has not  ↩
            compiled with SLURM library"  << std::endl;
        std::cerr << std::endl;
        exit(1);
        #endif
        batchType = SLURM;

        } else if (batchTypeStr == "LSF") {
        #ifndef HAVE_LSF
        std::cerr << std::endl;
        std::cerr << "Error: can't initialize LSF batch type because this server has not  ↩
            compiled with LSF library"  << std::endl;
        std::cerr << std::endl;
        exit(1);
        #endif
        batchType = LSF;
        } else {
        std::cerr << std::endl;
```

```
        std::cerr << "Error: invalid value for batch type parameter (must be 'TORQUE' or ' ↩
            LOADLEVELER' or 'SLURM' or 'LSF')"  << std::endl;
        std::cerr << std::endl;
        exit(1);
        }
```

La modification à ajouter dans le fichier **TMS/src/slave/slave.cpp** est le cas de test permettant d'initialiser le type du nouveau batch. Ci-dessous un exemple montrant les initialisations de quatres batchs (TORQUE, LOADLEVELER, SLUM et LSF).

```
        batchTypeStr = argv[2];
        if (batchTypeStr == "TORQUE") {
        batchType = TORQUE;
        } else if (batchTypeStr == "LOADLEVELER") {
        batchType = LOADLEVELER;
        } else if (batchTypeStr == "SLURM") {
        batchType = SLURM;
        } else if (batchTypeStr == "LSF") {
        batchType = LSF;
        } else {
        std::cerr << "Error: invalid value for batch type parameter (must be 'TORQUE' or ' ↩
            LOADLEVLER' or 'SLURM' or 'LSF')" << std::endl;
        throw UMSVishnuException(ERRCODE_INVALID_PARAM, "slave: invalid value for batch  ↩
            type parameter (must be 'TORQUE' or 'LOADLEVLER' or 'SLURM' or 'LSF')");
        }
```

Chapter 6

Modification de la classe SSHJobExec

La modification à ajouter dans la classe **TMS/src/server/SSHJobExec.cpp** est le cas de test correspondant au nouveau batch dans la méthode **SSHJobExec::convertBatchTypeToString()**. Ci-dessous le code de la méthode avec les batchs actuels (TORQUE, LOADLEVELER, SLURM et LSF).

```
std::string SSHJobExec::convertBatchTypeToString(BatchType batchType) {
std::string value;
switch(batchType) {
case TORQUE:
value = "TORQUE";
break;
case LOADLEVELER:
value = "LOADLEVELER";
break;
case SLURM:
value = "SLURM";
break;
case LSF:
value = "LSF";
break;
default:
value = "UNKNOWN_BATCH_TYPE";
break;
}
return value;
}
```

Eventuellement la méthode **SSHJobExec::sshexec** pourrait être modifiée si le nouveau batch à ajouter envoie ses erreurs sur la sortie standard. Sa modification n'est pas nécessaire si le nouveau batch offre une fonction permettant de récupérer ces erreurs sous forme de chaîne de caractères, et dans ce cas le développeur pourra envoyer cette chaîne de caractère comme exception vishnu dans la classe concrète du nouveau batch server.

Chapter 7

Modifications des CMakeFiles

Le développeur doit d'abord créer le fichier Find**NomDuNouveauBatch** dans le repertoire *Cmake* de la racine VISHNU. Les fichiers CMake à modifier sont : *CMakeLists.txt* de la racine, *TMS/src/CMakeLists.txt* et *TMS/src/config.cmake*.

Chapter 8

Résumé

Les modifications à faire pour l'ajout d'un nouveau batch dans TMS sont :

- Création des fichiers TMS/src/server/**NomDuNouveauBatch**Server.hpp et TMS/src/server/**NomDuNouveauBatch**Server.cpp
- Modification du fichier *TMS/src/sed/tmssed.cpp*
- Modification du fichier *TMS/src/slave/slave.cpp*
- Modification du fichier *TMS/src/server/SSHJobExec.cpp*
- Modification du fichier *TMS/src/server/BatchFactory.cpp*
- Création du fichier Find**NomDuNouveauBatch** dans le repertoire *Cmake de la racine VISHNU*
- Modification du fichier *CMakeLists.txt* de la racine VISHNU
- Modification du fichier *TMS/src/CMakeLists.txt*
- Modification du fichier *TMS/src/config.cmake*
- Modification des documents *TMS/design/docbook/TmsDesign-template.docbook, TMS/design/Archi.asta, TMS/design/data.asta et TMS/design/Design.asta*
- Modification des documents *TMS/test/testPlan/VISHNU_D4_1b-TMS-PlanTests.odt et TMS/test/testReports/reportTMSTest.docbook*
- Modification des documents *core/doc/usermanual/docbook/userman-template.docbook et core/doc/adminmanual/docbook/adminman template.docbook*
- Modification du document *core/specs/stb/STB_SoftHardWare.asta*