Гайд по использованию системы и написанию программ для исполнения

January 17, 2015

1 Использование системы

1.1 Запуск

ruby main.rb [n] [fabric_ name] [args]
где:

- *n* - параметр запуска системы
- *fabric_ name* - имя фабрики, которое будет использоваться системой для генерации фрагментов задач и фрагментов вычислений
- *args* - аргументы конструирования фабрики

  Замечания:

  - Имя фабрики должно совпадать с именем файла (fabric_name.rb) и, соответственно, с именем класса
  - Описание параметров запуска содержатся в FP-protocol.pdf

2 Написание программ для исполнения

Для написание программы для исполнения в системе необходимо создать свой класс фабрики на языке ruby. Требования для фабрики - наличие методов *generate_ data()*, *generate_ tasks()* и *restore_ task(id).*

- *generate_ data()* - генерирует массив объектов класса Data_
- *generate_ tasks()* - генерирует массив объектов класса Task
- *restore_ task(id)* - восстанавливает объект класса Task по идентификатору

  Объект фабрики конструируется с аргументами, заданными при запуске системы.

2.1 Необходимые знания об объектах системы

2.1.1 Класс Data_

Используемые поля для пользователя:

- *id* - идентификатор фрагмента данных
- *data* - контейнер для данных

  Данные должны представляться своим классом и иметь методы serialize() (сериализация в строку) и статичный метод new_deserialize(string) (для получения нового объекта по сериализованной строке). Описание этого класса может присутствовать в одном файле с фабрикой или в отдельном, но с добавлением зависимости (require) в файл с фабрикой.

  Описанные поля имеют методы доступа к ним извне только на чтение.

Конструирование объекта данных:

- *data = Data_.new(id, my_data_object)*

2.1.2 Класс Task

Используемые поля для пользователя:

- *dest_id* - номер узла, для которого назначается данный фрагмент вычислений
- *name* - имя фрагмента. Не имеет ничего общего с идентификатором фрагмента, можно использовать для отладки вычислений
- *data_deps* - массив идентификаторов фрагментов данных, необходимых для исполнения
- *priority* - приоритет исполнения фрагмента вычислений
- *execution_code* - исполнительный код фрагмента; объект типа Proc
- *id* - идентификатор фрагмента вычислений

  Следующие поля имеют методы доступа извне только на чтение: *dest_id*, *data_deps*, *priority*, *id*. С доступом на запись: *name*.

Используемые методы для пользования:

- *add_data_dep(data_id)* - метод добавления зависимости по данным для текущего фрагмента вычислений. (*data_id* - идентификатор данных)

Конструирование объекта вычислений:
task = Task.new(dest_id, id, execution_code, priority)