Запуск CluBORun

Для запуска CluBORun-а необходимо сделать следующее:

1. Получить бинарные файлы BOINC-клиента, работающие на узлах кластера (либо компиляцией клиента из исходных кодов, либо его скачиванием с сайта проекта, и т.д.) и поместить их каталог который мы назовём далее как <home>/node000 где <home> - каталог, в рамках которого и будет идти работа CluBORun-а;
2. Необходимо скомпилировать файл start\_boinc.cpp в бинарный файл start\_boinc;
3. В каталог <home> скопировать файлы catch\_node.awk, catch\_node.sh, start\_boinc;
4. В файле catch\_node.awk в 148 строке в предложении “startCommand = ….” исправить путь к файлу start\_boinc. В имеющемся примере эта строка выглядит так, как показано ниже, а исправляемый путь выделен красным:  
   startCommand = "mpirun VIADEV\_USE\_AFFINITY=0 -np " nodesInTask\*coresPerNode " -maxtime " catchNodeTime " /home2/mpc1/SAT/start\_boinc " taskName " " workPath " " startFlag " " stopFlag " " boincRunTime " " nodesList;
5. В файле catch\_node.sh, в третьей строке указать необходимый путь, задаваемый в составе LD\_LIBRARY\_PATH, если это необходимо:  
   export LD\_LIBRARY\_PATH=/home2/mpc1/SAT/lib:$LD\_LIBRARY\_PATH
6. В пятой строке задаём путь к каталогу <home>:  
   cd /home2/mpc1/SAT
7. Формируем экземпляры BOINC, которые мы будем запускать на узлах кластера. Это можно сделать тремя способами:  
   а) Самый простой – сделать копии BOINC-клиентов простым копированием node000 в node001, node002, …, nodeN;  
   б) Менее затратный – сделать каталог node001 в котором прописать символические ссылки на все файлы и каталоги из node000, а потом – размножить его в node002, …, nodeN;  
   в) Самый быстрый и оптимальный – скопировать в <home> каталог fabric, в подкаталоге matrix которого поместить символические ссылки на node000, после чего запустить файл create\_node.sh с двумя параметрами – началом и окончанием диапазона генерируемых узлов. Например:  
   create\_node.sh 1 32  
   И созданные скриптом каталоги вида nodeXXX перенести в <home>;
8. Создаём список задач, формируя файл <home>/all\_tasks.txt следующего формата:  
   <Название задачи> <название start-флага> <название stop-флага> <путь к каталогу <home>> <каталог экземпляра #1> <каталог экземпляра #2> …  
   Например:  
   task001 task001.start task001.stop /home2/mpc1/SAT node001 node002 node003 node004 node005 node006 node007 node008
9. Добавляем вызов catch\_node.sh в таблицу cron-а, задав нужные пути:  
   \*/5 \* \* \* \* /home2/mpc1/SAT/catch\_node.sh >> /home2/mpc1/SAT/catch\_node.log
10. После того, как задача запустилась – подключаем экземпляры BOINC к проекту воспользовавшись скриптом сделанным по подобию скрипта attach\_node\_sat.sh из каталога fabric, но с исправленными путями к <home>, заданием переменной LD\_LIBRARY\_PATH (если это вообще необходимо), URL-ом проекта и идентификатором участника. Вызывать скрипт необходимо с параметрами следующего вида:  
    attach\_node.sh <каталог экземпляра №1 (в задаче) > <название узла, на котором он запустился> <каталог экземпляра №2 (в задаче) > <название узла, на котором он запустился> …  
    Например:  
    attach\_node.sh node001 node12 node002 node73 node003 node154 …  
    Узнать какие экземпляры BOINC на какие узлы попали – можно в файлах формируемых mprirun-ом. После того, как экземпляры BOINC были подключены к проекту, более их подключать не надо.  
    Также можно воспользоваться weak account key, файл под именем account\_<URL-проекта>.xml с которым можно просто положить в каталог с экземпляром BOINC и который автоматически подхватится при его запуске. Например - account\_sat.isa.ru\_pdsat.xml в случае с SAT@home. Это вариант более простой, но если пароль участника, к которому приписываются эти экземпляры BOINC – будет сменён, то сменится и weak account key и эти экземпляры BOINC-клиентов не смогут получать задания.