MOE
Mini-howto

Alexis Rodríguez
{apokalexsys@gmail.com}

Requisitos

<> Linux
<> ssh
<> g++ 4.1.2, toolkit de desarrollo gnu para c++
<> mpich 1.2.7p1 http://www.mcs.anl.gov/mpi/mpich1/

Instalación

<> mpich

Si lo piensan usar paralelo pasen y vean[1] para configurar ssh para que no pida passwords, entre la máquina remota y la local usando vuestro usuario.

```
$./configure && make
```

Poner en el path el directorio bin del MPICH.

Si está todo bien:

```
$mpichversion
 MPICH Version:          1.2.7p1
 MPICH Release date:     $Date: 2005/11/04 11:54:51$
 MPICH Patches applied:  none
 MPICH configure:
 MPICH Device:           ch_p4
```

<> moe

Moe es el framework que va a ser producto de mi proyecto de grado, para instalarlo descompriman el tar.bz2
y editen el archivo $(MOE_HOME)/environment, paso a detallar

```
MOE_HOME = /* lo que se imaginan, acá va el directorio donde descomprimieron el tarball*/
MPICH_COMPILER = mpicxx /* el path al compilador de mpich, como lo pusimos en el
                         * path, alcanza con el nombre*/

MPICH_RUN = mpirun      /* idem pero éste es el runtime*/


LIB=$(MOE_HOME)/lib               /* directorios varios, si no cambian nada no hay porque tocarlos*/
DOC=$(MOE_HOME)/doc/informe
BIN=$(MOE_HOME)/bin
SRC=$(MOE_HOME)/src

ALGORITHMS_LIBRARY_NAME=moe-algorithms  /* nombres de las bibliotecas generadas*/
ENGINE_LIBRARY_NAME=moe-engine
PROBLEMS_LIBRARY_NAME=moe-problems

FLAGS = -g -pipe -W -Wall               /* flags para compilar y linkeditar, vean la ayuda de gcc si
tienen dudas*/

CXX = $(MPICH_COMPILER)
LINK = $(CXX) $(FLAGS) -L$(LIB) -l$(ALGORITHMS_LIBRARY_NAME) -l$(ENGINE_LIBRARY_NAME) -l
$(PROBLEMS_LIBRARY_NAME)
```

1 http://blogs.translucentcode.org/mick/archives/000230.html

```
IN = -o
CXXFLAGS = $(FLAGS) -I$(MOE_HOME)/src

ALGORITHMS=algorithms                   /* nombres de los "namespaces" */
ENGINE=engine
PROBLEMS=problems

LAUNCHER=launcher                       /* nombre del ejecutable*/

EXECUTABLE=$(BIN)/launcher              /* como se ejecuta */
```

Habiendo hecho esto estamos en condiciones de compilar en ${MOE_HOME}

```
$make rebuild
```

<> en el directorio ${MOE_HOME}/doc/html está la documentación doxygen de las clases, no es gran cosa pero da para arrancar

Uso

El framework se puede descomponer en tres "namespaces"

<> algorithms que tiene la implementación de NSGAII por ahora (les paso también el paper del algoritmo por si lo precisan)

<> engine son las clases que sirven de soporte de todo el resto, ahí están las definiciones interesantes, yo empezaría a vichar por ahi

<> problems ahi van a encontrar, las implementaciones de los problemas, incluyendo:

<> codificación de soluciones en reales y binarios (RealArraySolution y IntArraySolution)

<> operadores de selección, cruce, mutación y migración (vean el paquete engine y en particular MOEA)

<> problemas, por ahora los únicos problemas que tengo definidos son los ZDT que son un set de prueba para estas cosas.

<> launcher, acá es donde se pone la carne en el asador, un .cc que instancia y configura el framework para correr los problemas ZDT. Lo importante de todo esto me parece que puede ser el archivo de configuración, ${MOE_HOME}/bin/moea.properties está bastante claro pero les puede traer problemas, pregunten que no molestan.

Por lo que estuve mirando la idea de vuestro laboratorio es resolver un problema de scheduling, de pique van a tener que definir como codificar la solución, los operadores que van a usar, el problema (extendiendo las clases correspondientes) y como van a presentar los resultados, actualmente el framework, está sacando el histórico de las generaciones y además un archivo para octave[2] ustedes vean que les parece mejor, pero es un punto a tomar en cuenta desde el vamos.

Les paso también una versión super-alfa del informe del proyecto, que lamentablemente tiene poco que les sea útil pero les puede venir bien (¿?)

Seguramente a medida que vayan avanzando les surjan dudas no lo piensen dos veces, tírenme un mail

2 http://www.gnu.org/software/octave/