Philipps Universität Marburg

Prof. Dr. R. Loogen

HERIOT WATT UNIVERSITY

Summer School on Advances in Programming Languages 2014, Eden Lab Session, Thursday, August 21, 11am–12.30am

---

Working with the Eden system

General: For extended descriptions on Eden, the Eden compiler, the skeleton library, the trace viewer and additional material visit the Eden web pages:

http://www.informatik.uni-marburg.de/~eden

Eden programs have to import the module `Control.Parallel.Eden`.

Compile and run Eden programs (on the lab machines (multicores)): Use the compiler in the main DSG directory and copy the sample code from my local materials directory:

```
export PATH=/home/dsg/OPT/x86_64-unknown-linux/bin:$PATH
cp ~rita/AiPL/Materials .
```

Copy the file `~rita/.cabal/config` into your directory `~/.cabal` and then install the Eden skeleton library using `cabal install edenskel`. Look at

http://hackage.haskell.org/package/edenskel

for an overview of the skeleton library and the interfaces of the predefined skeletons.

It is worthwhile to define the following alias:

```
alias eden='ghc-7.6.2-eden-stage2 --make -O2 -eventlog -rtsopts'
```

Now you can compile e.g. the mandelbrot example like this:

```
eden -parcp  mandelbrot.hs
```

and run it like this

```
./mandelbrot 0 200 1 -out +RTS -N4 > out.ppm
```

This generates a mandelbrot picture in the file out.ppm which you can view like this:

```
display out.ppm
```

Eden programs use the RTS-options

- `-N<n>` to set the number of virtual machines (default is 1 VM per host)
- `-qQ<X>M` to set the size of the message buffer, and
- `-H<X>M` for the initial heap-size.

If a program crashes, you probably have to increase the message buffer.

Runtime-tracing of Eden programs: The above alias compiles programs with the option `-eventlog`. This allows to start a program with the runtime option `-ls` to create a runtime profile which can be visualized with the Eden trace viewer EdenTV.

see next page

Compile and run Eden programs on the beowulf cluster: Use `-parmpi` instead of `-parcp` when compiling Eden programs.

First, you will need to initiate MPD (the MPI demon) on the cluster, i.e. a file named `.mpd.conf` must be present in your home directory with read and write access only for you. This file must contain at least a line with: `MPD_SECRETWORD=<secretword>`. One way to safely create this file is to do the following:

```
cd $HOME
touch .mpd.conf
chmod 600 .mpd.conf
```

and then use an editor to insert a line like

```
MPD_SECRETWORD=mr45-j9z
```

into the file. Of course use some other secret word than mr45-j9z.

Moreover, it will be necessary to establish a password-less ssh-connection:

```
ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub <user>@jove.macs.hw.ac.uk
```

To run Eden programs on the **cluster** (i.e. more than one node) e.g. bwlf01-04 do the following:

1 list the nodes in a hostfile (e.g. `mpihosts`):

```
bwlf01
bwlf02
bwlf03
bwlf04
```

Use the runtime option `MPI@<hostfile>` to specify the hostfile, if you do not use the default `mpihosts`.

2 add current directory to path:

```
PATH=`pwd`:$PATH
```

3 start `mpd` (mpi process) with 4 nodes (i.e 8x4=32 processors in total):

```
mpdboot -n 4 -f mpihosts
```

Note: you might need to run `mpdallexit` first to exit existing running mpd instances. run program as shown above, specifying -N up to 32 processors.

Exercises

1 Divide and Conquer

easy

(a) Develop parallel versions of the program `nfib.hs` using the divide-and-conquer skeletons `disDC` and `flatDC` of the library
`Control.Parallel.Eden.DivConq` .

(b) Run your parallel programs on i machines, where $i \in \{1, 2, 4, ...\}$ (runtime option `-N`i).

(c) Use the Eden trace viewer to analyse the parallel program behaviours.

2 Parallel `map-reduce`

easy

(a) Use the skeletons `parMapRedl` and `offlineParMapRedl` of the library
`Control.Parallel.Eden.MapReduce`
to develop parallel versions of the program `sumEuler.hs`.

(b) Run your parallel programs on 8 machines (runtime option `-N8`) and analyse the parallel program behaviours using the Eden trace viewer EdenTV.

(c) Write an own skeleton `myParMapRedl` to improve work balance of the processes and check whether you were able to improve the parallel program behaviour. Compare the runtimes of the sequential program and your parallel version. Do you observe a speedup?

medium

(d) Use an appropriate composition of `parMap` and `parRed` (Module `parRed.hs` in `ExamplePrograms`) to parallelise the program `sumEuler.hs`. How does this version behave in comparison with the previous program versions?

3 Parallel Maps

medium

Write different parallel versions of the Julia-Set program `juliaSets.hs` using

(a) the `parMap` skeleton.

(b) the `farm` skeleton and the un-/shuffle function for task distribution.

(c) the `offlineFarm` skeleton and the un-/shuffle function for task distribution.

(d) the `workpoolSorted` skeleton

of the modules `Control.Parallel.Eden.Map` and `Control.Parallel.Eden.Workpool`, respectively. Control the number of messages using chunking of input and output lists of the processes.

Use EdenTV to analyse the runtime behavior of the different versions. Which skeleton is most appropriate for this problem?

4 Skeleton Composition and Iteration

advanced

(a) Write a parallel version of the conjugate gradient program `cg.hs` using the `iterUntil` skeleton from the module `Control.Parallel.Eden.Iteration`.

(b) Replace the `iterUntil` skeleton with an iteration of the `parMap` skeleton and remote data to establish a direct communication between the corresponding processes.

(c) Compare the runtime behaviours of both program versions using EdenTV.