SCOOP v2.0.0
Manual

October 23, 2013

Author: *Foivos S. Zakkak*
zakkak@ics.forth.gr

Foundation for Research and Technology - Hellas (FORTH)
Institute of Computer Science
N. Plastira 100
Vassilika Vouton, GR-700 13 Heraklion, Crete, Greece

License

Contents

1 Annotation Syntax

```
#pragma scoop start(list of variables)
#pragma scoop finish
#pragma scoop malloc
#pragma scoop free
#pragma scoop sync
#pragma scoop barrier
#pragma scoop wait all
#pragma scoop wait on(list of variables)
#pragma scoop task [in(<input parameters>)]
                   [inout(<input parameters>)]
                   [out(<input parameters>)]
```

Parameter notation:

Non stride: `<parameter>[\[number of elements (for arrays)\]]`

Stride: `<parameter>\[Block Rows|Block Columns\]\[[Array Rows|]Array Columns\]`
Array Rows is optional and is totally ignored

The parameter size/stride/els/elsz must be an expression, thus we don't allow function calls. Also there is no support for the conditional operator (? :)

Example: `#pragma scoop task in(a, b[4]) out(c[16])`

2 Installing

This section describes how to install SCOOP on your system. We suppose that you have checked out/cloned SCOOP under `/opt/scoop` directory. However the same instructions should apply for any alternative path, simply replacing `/opt/scoop` with the desired alternative path for the rest of this section.

2.1 Dependencies

In order to build SCOOP you will need to install the following packages:

- ocaml >= 3.11.2
- camlp4/ocaml-camlp4/ocaml-camlp4-devel
- flex
- bison
- indent

- ncurses-devel
- emacs
- gperf

2.2 Compile

To compile SCOOP you have to run `configure` and then `make`.

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

2.3 Install

You can install SCOOP running

Code
```
$   sudo make install
```

this will create a copy of the scoop executable in `/usr/local/bin`.
NOTE: You still have to keep the current directory to your system.

Alternatively you can append `/opt/scoop` to the `PATH` variable.
i.e.

Code
```
$   echo "export PATH=$PATH:/opt/scoop" >> $HOME/.bashrc
```

2.4 Uninstall

You can uninstall SCOOP running

Code
```
$   sudo make uninstall
```

this will erase the copy of the scoop executable from `/usr/local/bin`.

If you chose the alternative method of adding `/opt/scoop` to your `PATH` variable, simply remove the added line from your `.bashrc`

3 Usage

Code
```
$   scoop --runtime=<myrmics/dummy> [options] <file> [file2 ...]
```

3.1 Options

`--runtime`	Define the target runtime/architecture myrmics | dummy
`--cflags`	Defines the flags you want to pass to gcc
`--include-path`	Defines the path containing the runtime header files.
`--debug-SCOOP`	Print debugging information
`--trace`	Trace SCOOP
`--out-name`	Specify the output files' prefix. e.g. (default: scoop_trans) will produce scoop_trans.c
`--pragma`	Specify the string constant following the pragma e.g. (default: runtime's name). For myrmics will recognise `#pragma myrmics`
`--disable-sdam`	Disable the static dependence analysis module

4 Extending SCOOP

In order to add support for your runtime on the SCOOP compiler you have to take the following steps.

1. copy src/scoop_dummy.ml and src/scoop_dummy.mli to src/scoop_myruntime.ml and src/scoop_myruntime.mli respectively.
2. append scoop_myruntime to the SCOOP_MODULES variable in Makefile.
3. Perform any required changes to src/scoop_myruntime.ml
4. Append `Scoop_myruntime.options` to `fd_extraopt` in src/scoop.ml
5. Add the following lines to `match !arch with` in src/scoop.ml
```
| "myruntime" ->
  new Scoop_myruntime.codegen callgraph !gen_file !pragma_str !includePath
```

5 Common Errors/Limitations/Known Bugs

- Adding a semicolon at the end of `#pragma`s will make SCOOP fail
  i.e. `#pragma scoop sync;`
- Putting `#pragma scoop barrier` at the end of a block will make SCOOP fail (add a semicolon right below the `#pragma` to fix it).
- **Fatal error: exception Invalid_argument("Unknown")** you probably have wrong argument at a call tagged with `#pragma scoop task`
- Using DEFINES or MACROS in pragmas (preprocessor doesn't process them)

- Putting `#pragma`s directly above a declaration of a variable (pragmas are only supported above statements)
- Using directly the runtime API instead of the corresponging `#pragma` may result in SDAM not working properly.
- **Error: ”segment___0” not found in the #pragma scoop task** usually means that the tool is renaming a variable due to previous declaration try renaming it manually (e.g. segment2) (This should be fixed by now)