Dokumentation zum Programm iopattern2dios

Daniela Koudela
ZIH
Technische Universität Dresden

19. März 2013

Inhaltsverzeichnis

1 Wofür ist das Programm gut?

Das Programm *iopattern2dios* nimmt die Ausgabe von *iopattern/rabbit* und wandelt sie in ein XML-Format um, welches als Eingabe für das Programm *DIOS* verwendet werden kann.

1.1 Was macht das Open Trace Format (OTF)

Mit dem *Open Trace Format* (OTF) werden Trace-Informationen während dem Laufen von massiv parallelen Programmen gesammelt, um das Programmverhalten mittels

geeigneter Analyseprogramme detailliert untersuchen zu können.[1]

1.2 Was macht iopattern/rabbit?

Das Programm *iopattern/rabbit* analysiert und klassifiziert I/O-Aufrufe innerhalb einer OTF-Datei.

Um das Programm *iopattern/rabbit* zu erhalten, schickt man am besten eine entsprechende Anfrage an Michael Kluge (`michael.kluge@tu-dresden.de`).

1.3 Was macht DIOS?

DIOS ist ein in C++ geschriebener Dateisystem-Benchmark, um I/O-Aktivitäten einer Anwendung zu wiederholen Die Input-Datei muss im XML-Format sein. Diese beschreibt, welche I/O-Aktivitäten jeder Prozess ausführt.[2]

Um das Programm *DIOS* zu erhalten, schickt man am besten eine entsprechende Anfrage an Michael Kluge (`michael.kluge@tu-dresden.de`).

2 Installation

Damit man eine Trace-Datei in XML umwandeln kann, muss man OTF und *iopattern/rabbit* installieren. Hat man bereits die Ausgabe von *iopattern/rabbit*, so ist keine Installation der beiden Programme nötig.

iopattern2dios ist ein python-Programm und muss somit nicht groß installiert werden. Es reicht, die Datei `iopattern2dios.py` in ein geeignetes Verzeichnis zu legen.

2.1 Installation von OTF

Zuerst muss OTF-1.11 aus dem Internet (http://www.tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/software_werkzeuge_zur_unterstuetzung_von_programmierung_und_optimierung/otf) heruntergeladen werden. Man entpackt es in einem geeigneten Verzeichnis. Zusätzlich braucht man noch das devel-Paket der libz. Dazu muss das Paket `zlib1g-dev` installiert werden. Dann wechselt man in das neu entstandene Verzeichnis `OTF-1.11.2goldfish` und installiert das Programm entsprechend den Anweisungen aus der Datei `INSTALL`. Dabei muss beachtet werden, dass die Anweisung `make install` durch `sudo make install` ersetzt werden muss, damit es funktioniert.

[1] Für weitere Informationen zu OTF siehe http://www.tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/software_werkzeuge_zur_unterstuetzung_von_programmierung_und_optimierung/otf

[2] Weitere Details sind in der Dokumentation von *DIOS* erklärt. Diese befindet sich in der *DIOS*-Installation im Verzeichnis `docu/latex`.

2.2 Installation von iopattern/rabbit

Man folgt der Anleitung, die sich in der Datei `INSTALL` im Verzeichnis `iopattern` befindet. Eventuell muss man anstatt `./configure`

```
./configure --with-otf-root="/usr/local"
```

eingeben, um *iopattern/rabbit* zu sagen, wo sich OTF befindet.

3 Benutzung

3.1 Benutzung von iopattern/rabbit

Als Eingabe braucht *iopattern/rabbit* ein von *VampirTrace* erzeugtes Trace. Dieses besteht aus einer globalen Datei mit der Endung `.otf`, einer Definitionsdatei mit der Endung `.def.z` und für jeden Prozeß eine Datei, die die mit *libz* komprimierte Programmspur im OTF-Format enthält.

Um das Programm *iopattern/rabbit* zu starten, ruft man das Programm *rabbit* im Verzeichnis `iopattern/rabbit` auf und übergibt diesem die globale OTF-Datei.

Zum Beispiel:

```
~/iopattern/rabbit> ./rabbit --anonptr -b /home/userxy/testtrace/iot.otf
```

Als Ausgabe erhält man dann die folgenden 3 Dateien:

- Eine Datei namens `raw.dot`.
- Eine Datei namens `cleaned.dot`. Im Vergleich zu `raw.dot` sind hier redundante Synchronisationsknoten entfernt.
- Informationen, die nach `stdout` geschrieben werden.

Kurzanleitung von iopattern/rabbit

```
rabbit   -  analyze/classify I/O call within an Vampirtrace

Options:
 -h,--help                 show this help message
 -m,--monitor <directory>  monitor only files in this directory
                           can be given more than once
 -b,--barrier              sync I/O operations with blocking MPI events
 --mask                    put mask over generated tokens before handing
                           it over to the compression algorithm
 --dumpall                 dump all generated tokens
 --anonptr                 use anonymous pointers where appropriate
 --nocompress              don't compress graph during construction
 --graphgen                dump dot file after each generation step
```

```
  -g,--group <name>         sets the function group to be inspected,
  -f,--func <name>          sets the function name to be inspected,
                            option can be given more than once!
  --plaindump               dumps the raw data for each call
```

3.2 Benutzung von iopattern2dios

Kurzanleitung

```
Usage: python iopattern2dios.py [options]

Options:
  -h, --help                            show this help message and exit
  --encoding=ENCODING                   Encoding of the XML-file.
  -v, --verbose                         Turn verbosity on.
  -f INPUTDATA, --file=INPUTDATA        Path to the input-file.
  -o OUTFILE, --outfile=OUTFILE         Path to the output-file.

The default encoding is ISO-8859-1.

INPUTDATA must be a file, which contains the information, which is written
to stdout when running iopattern.

The format of OUTFILE is XML.
```

Ausführliche Anleitung

Das Programm *iopattern/rabbit* liefert drei verschiedene Ausgaben:

- Eine Datei namens `raw.dot`.
- Eine Datei namens `cleaned.dot`. Im Vergleich zu `raw.dot` sind hier redundante Synchronisationsknoten entfernt.
- Informationen, die nach `stdout` geschrieben werden.

Die beiden `*.dot`-Dateien werden von *iopattern2dios* nicht benötigt, sondern nur das, was nach `stdout` geschrieben wird. Diese Ausgabe muß als Datei vorliegen, zum Beispiel durch Umleitung von `stdout` in eine Datei beim Ausführen von *iopattern/rabbit*.

Zum Ausführen von *iopattern2dios* übergibt man die Datei `iopattern2dios.py` dem Programm Python:

```
python iopattern2dios.py
```

Befindet man sich nicht in dem Verzeichnis, in dem die Datei `iopattern2dios.py` liegt, so muß man die Datei mit dem vollständigen Pfad übergeben.

Werden keine Optionen angegeben, so geht *iopattern2dios* davon aus, daß die Ausgabe des Programms *iopattern/rabbit* in der Datei `input` gespeichert ist, die sich im aktuellen

Verzeichnis befindet. Die Ausgabe von *iopattern2dios* wird dann in der Datei `out.xml` im selben Verzeichnis mit dem Encoding ISO-8859-1 gespeichert.

Einen Überblick über die möglichen Optionen von *iopattern2dios* erhält man, wenn man das Programm mit der Option `-h` bzw. `--help` aufruft.

```
python iopattern2dios.py --help
```

Folgende Optionen sind möglich:

Langform	**Kurzform**	**Erklärung**
`--help`	`-h`	Zeigt die in der Kurzanleitung dargestellte Übersicht über die möglichen Optionen.
`--encoding=<Kodierung>`		Gibt die Kodierung der XML-Datei, die die Ausgabe des Programms darstellt, an. Ist keine Kodierung angegeben, so wird die Kodierung ISO-8859-1 verwendet.
`--verbose`	`-v`	Bei Verwendung dieser Option wird *iopattern2dios* gesprächig und teilt mit, welche Optionen gewählt wurden, welche Datei als Eingabe-Datei verwendet wird, in welche Datei die Ausgabe geschrieben wird, welche Kodierung benutzt wird und welche Zeilen aus der Eingabe-Datei nicht verarbeitet wurden.
`--file=<Eingabe-Datei>`	`-f <Eingabe-Datei>`	Mit dieser Option kann der Name der Eingabe-Datei von `input` auf einen beliebigen Wert geändert werden. Liegt die Eingabe-Datei in einem anderen als dem aktuellen Verzeichnis, so gibt man vor dem Dateinamen den vollständigen Pfad an.

`--outfile=<Ausgabe-Datei>`	`-o <Ausgabe-Datei>`	Mit dieser Option kann der Name der Ausgabe-Datei von `out.xml` auf einen beliebigen Wert geändert werden. Liegt die Ausgabe-Datei in einem anderen als dem aktuellen Verzeichnis, so gibt man vor dem Dateinamen den vollständigen Pfad an.

Die Ausgabe-Datei hat das richtige Format, um als Eingabe-Datei für das Programm *DIOS* zu fungieren.

4 Wichtige Hinweise

iopattern2dios ist in der Programmiersprache Python geschrieben. Die in der Datei `dios.dtd` definierten Tags und Attribute sind in *iopattern2dios* implementiert. Es ist möglich und auch wahrscheinlich, dass die Ausgabe von *iopattern/rabbit* Funktionen enthält, die nicht in `dios.dtd` definiert sind. In diesem Fall wird eine Warnung auf `stdout` geschrieben. Ferner nimmt *iopattern2dios* an, dass das Argument jeder Funktion in der Eingabe-Datei eine durch Komma getrennte Liste aus Schlüssel-Wert-Paaren ist, bei denen die einzelnen Schlüssel-Wert-Paare durch einen Doppelpunkt den Schlüssel vom Wert trennen. Dies ist jedoch nicht immer der Fall. So kann die Ausgabe von *iopattern/rabbit* zum Beispiel die Funktion

```
barrier_sync( 4)
```

enthalten. Wie man sieht, ist hier das Argument kein Schlüssel-Wert-Paar. Auch ist die Funktion `barrier_sync` nicht in `dios.dtd` definiert. In diesem Fall verwandelt *iopattern2dios* die Information wie folgt in XML:

- Die Funktion `barrier_sync` wird zu einem Tag.
- Die 4 wird als Wert dem Attribut `"unknown"` zugeordnet.

⇒ `<barrier_sync unknown="4" />`

Des Weiteren werden zwei Warnungen nach `stdout` geschrieben:

```
Warning: 4 is not a key-value object. Can not find delimiter ':' in this string.
Warning: The function barrier_sync is not defined in dios.dtd!
```

Durch die nach `stdout` geschriebenen Warnungen ist man immer informiert, welche Informationen der Eingabe-Datei nicht in `dios.dtd` definiert sind und eventuell Probleme bereiten könnten. Warnungen werden übrigens auch dann ausgegeben, wenn die Verbose-Option nicht benutzt wird.

Mapping

Dateien werden in einem bestimmten Modus geöffnet, z. B. lesend oder schreibend. Üblicherweise wird dieser Modus durch einen Buchstaben (z. B. r oder w) angegeben. In der Ausgabe von *iopattern/rabbit* wird dieser Modus jedoch durch Zahlen kodiert, da dies *VampierTrace* so macht. Diese Zahlen müssen wieder in Buchstaben zurückverwandelt werden. Dies ist in *iopattern2dios* nach folgendem Schema implementiert:

1	r
2	w
4	a

Enthält die Eingabe-Datei an der entsprechenden Stelle eine andere Zahl als die in der obigen Tabelle erwähnten, so wird eine Warnung nach `stdout` geschrieben.

5 Weitere in diesem Zusammenhang möglicherweise nützliche Programme

- Um sich die Struktur und den Inhalt einer XML-Datei gut anschauen zu können, ist das Programm *treeline* hilfreich.
- Um sich den Inhalt der beiden `*.dot`-Dateien, die *iopattern/rabbit* als Ausgabe liefert, anschauen zu können, ist das Programm *Graphviz* hilfreich.