Addendum

zum Artikel

„Aufgedreht – Kernel Tuning“

Linux Magazin 07/04

Dipl.-Inform. Ingo A. Kubbilun

www.pyrillion.org

Version 0.1, 06/01/2004

0. Inhaltsverzeichnis

0. INHALTSVERZEICHNIS....2

0.1 Dokumentenversion....3

1. PATCHEN UND ÜBERSETZEN DES KERNELS....4

1.1 Unterstützte Kernel-Versionen....4
1.2 Entpacken des Kernels und des Patch-Archivs....4
1.3 Download und Installierung des Intel® Compilers....4
1.4 Übersetzung und Installierung der Tools....4
1.5 Patchen des Kernels....5
1.6 Konfiguration und Übersetzen des Kernels....5
1.7 Installierung des Kernels....5
1.8 Troubleshooting....5

2. PROFILGELEITETE OPTIMIERUNG DES KERNELS....6

2.1 Vorbereitung des Kernel-Moduls....6
2.2 SMP-Kernel (symmetric multi-processing)....6
2.3 Phase 1/3: PGO-Instrumentalisierung des Kernels....6
2.4 Phase 2/3: Generierung des dynamischen Profils....7
2.5 Phase 3/3: Feedback-Übersetzung des Kernels....7
2.6 Ausdehnung des Profilings auf weitere Kernel-Bereiche....8
2.7 PGO-Instrumentalisierung von Kernel-Modulen....8
2.8 Erstellung der HTML-Darstellung des Profils....8

0.1 Dokumentenversion

Version	Datum	Autor(en)	Bemerkung
0.1	06/01/2004	Ingo A. Kubbilun	Erste Fassung

1. Patchen und Übersetzen des Kernels

1.1 Unterstützte Kernel-Versionen

2.6.3, 2.6.4, 2.6.5, 2.6.6

Download: http://www.kernel.org/pub/linux/kernel/v2.6/

Das „**x**" steht im folgenden für eine der Versionen 3, 4, 5 oder 6.

1.2 Entpacken des Kernels und des Patch-Archivs

```
cd /usr/src
```

```
tar xfzp linux-2.6.x-tar.gz   oder:   tar xfjp linux-2.6.x-tar.bz2
```

```
tar xfzp linux-2.6-icc-0.9.tar.gz   oder:   tar xfjp linux-2.6-icc-0.9.tar.bz2
```

1.3 Download und Installierung des Intel® Compilers

Sofern Sie den Intel® C/C++ Compiler for Linux nicht kommerziell erworben haben, können Sie den Compiler zur ausschließlich privaten Nutzung in der *„free non-commercial unsupported version*" benutzen:

Download: http://www.intel.com/software/products/compilers/clin/noncom.htm

Führen Sie die Registrierungsprozedur durch, und laden Sie das Archiv herunter. Des weiteren erhalten Sie von Intel® eine E-Mail mit der Lizenzschlüsseldatei. Installieren Sie den Compiler nach Intel®'s Anweisungen. Die Installation ist skript-gesteuert.

Während der Installation werden Sie nach der Lizenzschlüsseldatei gefragt. Halten Sie diese bereit. Setzen Sie die Umgebungsvariablen `PATH`, `MANPATH` und `LD_LIBRARY_PATH` auf die entsprechenden Intel®-Verzeichnisse.

Wenn Sie aus Sicherheitsgründen die Variable LD_LIBRARY_PATH nicht modifizieren möchten, so können Sie alternativ auch die Datei `/etc/ld.so.conf` ändern und den Linker neu konfigurieren (über `ldconfig`).

1.4 Übersetzung und Installierung der Tools

Für das Übersetzen der Tools benötigen Sie den GNU C/C++ Compiler ab 2.95. Alternativ können Sie auch die Umgebungsvariable `CC` auf `icc` setzen, um mit dem Intel® Compiler zu übersetzen:

```
cd /usr/src/linux-2.6-icc-0.9
```

```
./configure
```

```
make && make install
```

Dies übersetzt und installiert die folgenden Programme bzw. man pages:

- C compiler delegate `ccd` (in `/usr/local/bin`)
- linker delegate `lkd` (in `/usr/local/bin`)
- PGO daemon `pgod` (in `/usr/local/sbin`)

- man page `ccd(1)` (in `/usr/local/man`)
- man page `lkd(1)` (in `/usr/local/man`)
- man page `pgod(8)` (in `/usr/local/man`)
- man page `intlpgo(9)` (in `/usr/local/man`, nähere Informationen zum kernel PGO)
- man page `lkpatch(9)` (in `/usr/local/man`, nähere Informationen zum kernel patch)

1.5 Patchen des Kernels

```
cd /usr/src/linux-2.6.x
patch -Np1 < ../linux-2.6-icc-0.9/patches/patch-2.6.x-ICC
```

1.6 Konfiguration und Übersetzen des Kernels

```
export CCC="icc"
export CCC_OPTS="—no-common --remove-force-junk --quiet"
make xconfig  oder:  make menuconfig
make bzImage && make modules
```

Hinweis: Konfigurieren Sie für die ersten Gehversuche **nicht** das PGO-Feature (Profile Guided Optimization, vgl. Abschnitt 2).

1.7 Installierung des Kernels

```
make modules_install
cp System.map /boot/System.map-2.6.xICC
cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.xICC
```

Legen Sie für den Intel®-Kernel eine neue Sektion in der Datei `/etc/lilo.conf` an und konfigurieren Sie anschließend den Bootloader LILO über `/sbin/lilo`. Das komprimierte Image `vmlinuz-2.6.xICC` liegt im oben gezeigten Beispiel in `/boot` vor.

1.8 Troubleshooting

Überprüfen Sie bei Problemen die folgenden Punkte:

- Verfügbarkeit der GNU binutils-2.14 oder binutils-2.15 (z.B. `as --version`)
- Version und Verfügbarkeit des Intel® C/C++ Compilers for Linux (z.B. `icc -V`); der Compiler muss in der Version 8.0.055 (oder höher) vorliegen
- Verfügbarkeit der Übersetzertools (z.B. `man ccd`)

2. Profilgeleitete Optimierung des Kernels

2.1 Vorbereitung des Kernel-Moduls

Bevor der Linux-Kernel dynamische Profilinformationen generieren kann, muss das Kernel-Modul `intlpgo` um den originalen Intel®-PGO-Code bereichert werden. Gehen Sie dazu folgendermaßen vor:

```
cd /usr/src/linux-2.6-icc-0.9/ExtractPGO
```

```
make KRNLSRCDIR=/usr/src/linux-2.6.x
```

Hierdurch wird u.a. das `Makefile` in `/usr/src/linux-2.6.x/arch/i386/intlpgo` modifiziert sowie die neue Datei `intlPGO.object` angelegt, gegen die das Kernel-Modul nun gelinkt wird.

Beachten Sie bitte, dass das modifizierte Kernel-Modul zum „Verschmutzungszustand" (*taint status*) P führt, da es proprietären Intel®-Code enthält. Einen derartig modifizierten Kernel dürfen Sie keinesfalls verteilen. In der dritten Phase der Feedback-Übersetzung (vgl. Abschnitt 2.5) wird das PGO-Kern-Modul automatisch deaktiviert. Der resultierende (und optimierte) Kernel enthält daher keinen Intel®-Code mehr.

2.2 SMP-Kernel (symmetric multi-processing)

Die PGO-Instrumentalisierung ist auf SMP-Kernel anwendbar. Sofern die Erzeugung der dynamischen Profildaten eines SMP-Kernels auf einem Einprozessor-Rechner erfolgt, sind die dynamischen Profildaten akkurat.

Die Verkettung der PGO-Segmentpakete[1] zu einer Liste ist auch auf Multiprozessor-Maschinen sicher. Da die verwendeten Zähler 64 Bit breit sind, und es zu deren Inkrementierung auf Intel®-Prozessoren mehrerer Prozessorbefehle bedarf, kann es beim Update der PGO-Segmentpakete zu Ungenauigkeiten kommen. Dieses (Fehl-)Verhalten ist auf Multiprozessor-Maschinen nicht korrigierbar, da der Intel®-Compiler während der Zähleraktualisierung keine Synchronisationsmechanismen (z.B. *spinlocks*) einsetzt.

2.3 Phase 1/3: PGO-Instrumentalisierung des Kernels

Konfigurieren Sie den Kernel erneut und aktivieren Sie PGO im Hauptmenü. Es öffnet sich ein Untermenü mit verschiedenen Kernel-Bereichen. Selektieren Sie die Bereiche, für die Sie dynamische Profilinformationen erzeugen lassen möchten.

Da die PGO-Instrumentalisierung das Kernel-Image stark aufbläht, ist die gleichzeitige Selektierung aller Untermenüpunkte nicht möglich, da der Kern ansonsten nicht erstellt werden kann bzw. später nicht bootet. Sie können die Schritte in den Abschnitten 2.3, 2.4 und 2.5 für weitere Kernel-Bereiche wiederholen.

Beispiel (vier Kernel-Bereiche): `arch/i386/kernel`, `kernel`, `arch/i386/mm`, `mm`

Erstellen und Installieren Sie den Kernel und seine Module wie unter 1.6 und 1.7 gezeigt. Nach dem Übersetzungsvorgang muss sich die Datei `pgopti.spi` im Verzeichnis `/`

[1] In den PGO-Segmentpaketen werden die dynamischen Profildaten im RAM bis zur Speicherung auf die Festplatte vorgehalten.

`usr/src/linux-2.6.x` befinden. Sie enthält die statischen Profilinformationen, die der Intel® Compiler während der ersten Phase der Übersetzung generiert hat.

Legen Sie ein dediziertes Verzeichnis zur späteren Ablage der dynamischen Profildaten an, z.B. `/usr/src/profdata`.

2.4 Phase 2/3: Generierung des dynamischen Profils

Booten Sie den PGO-instrumentalisierten Kern, und starten Sie den PGO-Daemon, z.B.

```
pgod --start --prof-dir=/usr/src/profdata
```

Der PGO-Daemon schreibt nun alle 30 Sekunden ein dynamisches Profil auf die Festplatte. Die man page `pgod(8)` erklärt alle Optionen des Daemons.

Arbeiten Sie nun mit dem Linux-Kernel in genau der Art und Weise, wie er später zum Einsatz kommen soll. Abschließend stoppen Sie den PGO-Daemon:

```
pgod --kill
```

und booten den Linux-Kernel Ihrer gewohnten Arbeitsumgebung (nicht PGO-instrumentalisiert!).

2.5 Phase 3/3: Feedback-Übersetzung des Kernels

Zunächst müssen die gesammelten dynamischen Profildaten zu einer Datei zusammengefügt werden. Hierzu dient das Werkzeug `profmerge`, das zusammen mit dem Intel® Compiler installiert wird:

```
cd /usr/src/profdata
profmerge
```

Es resultiert die Datei `pgopti.dpi` mit dem dynamischen Profil des Linux-Kernels. Kopieren Sie nun die PGO-Dateien und die Kernel-Konfiguration an einen anderen Ort:

```
cd /usr/src
mkdir pgodata
cp ./linux-2.6.x/pgopti.* ./pgodata
cp ./linux-2.6.x/.config ./pgodata
cp ./profdata/pgopti.dpi ./pgodata
```

Bereinigen Sie nun den Kernel-Quellcodebaum, restaurieren Sie die PGO-Dateien und die Kernel-Konfiguration und führen Sie die Feedback-Übersetzung des Kernels durch:

```
cd /usr/src/linux-2.6.x
make mrproper
cp ../pgodata/* .
make bzImage && make modules
```

Das Vorhandensein der PGO-Dateien `pgopti.spi` und `pgopti.dpi` veranlasst den Erstellungsprozess nun, den Kernel nicht zu instrumentalisieren (PGO), sondern die vorhandenen PGO-Dateien zur Optimierung des Kernels einzusetzen.

2.6 Ausdehnung des Profilings auf weitere Kernel-Bereiche

Löschen Sie die Datei `pgopti.dpi` in `/usr/src/linux-2.6.x` aber **keinesfalls** `pgopti.spi`. Letztere wird bei neuen Übersetzungsvorgängen durch den Intel®-Compiler aktualisiert. Konfigurieren Sie nun den Kernel, um andere Bereiche des Kernels zu instrumentalisieren.

In Phase zwei (Abschnitt 2.4) erzeugt der PGO-Daemon nun dynamische Profildaten für die aktuell instrumentalisierten Teile des Kernels. Die ursprünglichen Profildaten in `/usr/src/profdata` werden dabei **nicht** überschrieben.

Löschen Sie die Datei `pgopti.dpi` in `/usr/src/profdata` und führen Sie das Werkzeug `profmerge` erneut aus. Es erzeugt eine aktualisierte Datei `pgopti.dpi` mit den dynamischen Profilen **aller** bisher gesammelten Daten, die nun in Phase 3 (Abschnitt 2.5) zur Optimierung des Kernels herangezogen werden.

Die PGO-Instrumentalisierung ist auf sämtliche Bereiche der Kernels anwendbar, auch auf jene, die im Menü der Kernel-Konfiguration nicht vorgesehen sind. Dazu müssen Sie die entsprechenden Makefiles manuell in den Phasen eins und drei modifizieren:

Phase 1: `EXTRA_CFLAGS += -prof_genx`

Phase 3: `EXTRA_CFLAGS += -prof_use`

2.7 PGO-Instrumentalisierung von Kernel-Modulen

Die PGO-Instrumentalisierung und die Anfertigung von dynamischen Profildaten ist auch auf Kernel-Module anwendbar. Das Profiling beginnt, nachdem das entsprechende Kernel-Modul geladen wurde.

Durch die interne Verkettung der PGO-Segmentpakete zu einer Liste (siehe auch Abschnitt 2.2) würde das Entfernen des Kernel-Moduls zu schweren Ausnahmen im Kernel führen. Der Kernel Patch modifiziert die Modul-Laderoutine des Kernels für den Fall des Entfernens von Modulen wie folgt:

1. Die aktuelle PGO-Segmentkette wird im RAM zwischengespeichert (Ring-Buffer).
2. Die PGO-Segmentkette wird zerstört.
3. Das Modul wird aus dem RAM entfernt.
4. Die PGO-Segmentkette wird automatisch neu erstellt.

Der Ring-Buffer im Kernel-Modul `intlpgo` hält das dynamische Profil bis zur nächsten Speicheraufforderung durch den PGO-Daemon `pgod` vor.

2.8 Erstellung der HTML-Darstellung des Profils

Das Intel®-Werkzeug `codecov` (vgl. man page) erzeugt aus den PGO-Profildateien einen Baum von HTML-Dateien mit einer farblich unterschiedlichen Darstellung der Code-Blöcke je nach dem Ausführungsverhalten des Kernels in Phase zwei (Abschnitt 2.4).