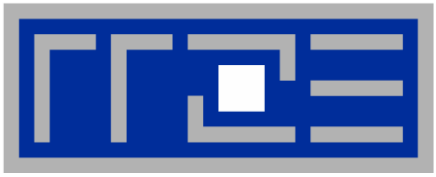


Intel VTune for Linux

Georg Hager
RRZE

14.07.2004

Outline

- **Intel VTune**
  - **Grundlagen: VTune für Linux und Windows**
  - **IP und Callgraph sampling**
  - **Profiling auf Maschinen-, Modul- und Zeilenebene**
  - **Callgraph viewer**
  - **Performance Counter sampling**
  - **Bits & Pieces**
- **Kurze Bemerkungen zu OProfile**

VTune-Grundlagen

- Eigentlich zwei Produkte (getrennt lizenziert)
  - VTune für Windows: GUI-Profiling-Tool mit „allem drum und dran“
  - VTune für Linux (vtl) 2.0: (Fast) reines Kommandozeilen-Tool
- Möglichkeiten
  - Sampling (verschiedene Metriken) von Systemebene bis auf Zeilen- bzw. Assemblerebene
  - Callgraph
- Kopplung beider Welten auf verschiedene Arten möglich
  - Steuerung eines Linux-„Samplers“ von der Windows-Oberfläche aus (Sampler ist Teil von VTune for Windows)
  - Einpacken der vtl-Samplingdaten und Visualisierung mit VTune for Windows
- Wichtig: VTune ist (noch) ein one-user-at-a-time-Tool!

Activities

- Basiseinheit fürs Profiling: „Activity“
  - Enthält kompletten Satz von Parametern (welches Programm, welche Metrik, Randbedingungen etc.)
  - Einrichten einer Activity mittels

    ```
    vtl activity [NAME] –c collector –app exec[,args]
    ```

  - Beispiel:

    ```
    vtl activity OPTIMIZED –c sampling –app ./a.out
    ```

  - Jede Activity bekommt einen eindeutigen Namen (abgesehen vom angegebenen) a*N* mit N=1,2,3,…
- Mehrere Activities pro Projekt sind möglich
  - Anzeigen von Activities: `vtl show [activity] [-a]`
  - Löschen von Activities: `vtl delete activity` oder `vtl delete –a[ll]`

Data Collectors

- Data Collectors
  - Mittels `-c` beim Einrichten der Activity angeben
- „`sampling`“
  - IP sampling mittels Counter Overflow
  - Alle Performance-Counter verwendbar
  - Liste mit `vtl query -c sampling`
  - Zeilenbasiertes Sampling bis auf Assembler-Ebene
  - Standardcounter auf IA32: `Clockticks`, `Instructions Retired`
- „`callgraph`“
  - Executable wird vor dem Lauf instrumentiert
    - Spürbarer Slowdown
  - Gprof-artiges Protokoll mit self-time und total time für jeden Call
  - GUI zur Auswertung vorhanden

Starten der Applikation

- Starten der Applikation mit den in der Activity festgelegten Parametern:

  ```
  vtl run [activity]
  ```

- VTune erkennt automatisch Inkompatibilitäten bei der gleichzeitigen Messung verschiedener Metriken und startet die Applikation u.U. mehrfach:

  ```
  vtl activity -c sampling -c callgraph ...
  ```

- Funktioniert auch bei nicht gleichzeitig verwendbaren Performance Countern
- Mehrere Runs pro Activity möglich

Beispiel für IP sampling

- Anwendung: PIC (proton-induced electron cooling)
- Variablen setzen

```
$ export PATH=/opt/intel/vtune/bin:$PATH
$ export VTUNE_USER_DIR=/tmp/user/vt
```

- Activity einrichten mit sampling collector:

```
$ vtl activity -c sampling -app ./pic.ia32
```

- Anzeigen der Activity:

```
$ vtl show -a
VTune(TM) Performance Analyzer 2.0 for Linux* Update 1
Copyright (C) 2000-2004 Intel Corporation. All rights
reserved.

a1__Activity1
$
```


Beispiel für IP sampling

- Starten der Applikation unter Kontrolle von VTune:

```
$ vtl run
Copyright (C) 2000-2004 Intel Corporation. All
rights reserved.

The Activity is running.
Mon Jul 12 17:01:02 2004 storm (Run 0) The Sampling
Collector is collecting samples based on the
following event(s): Instructions Retired,
Clockticks.
[... Programm-Ausgaben ...]
Mon Jul 12 17:02:22 2004 storm (Run 0) Sampling
data was successfully collected.
The Activity has finished running.
$
```

Beispiel für IP sampling

- **show zeigt jetzt, welche Daten gesammelt worden sind**

```
$ vtl show -a
VTune(TM) Performance Analyzer 2.0 for Linux* Update 1
Copyright (C) 2000-2004 Intel Corporation. All rights
reserved.

a1__Activity1
    r1_____Sampling Results [storm] - Mon Jul 12
17:02:22 2004
    r2_______Run 0
    r3_________Instructions Retired
    r4_________Clockticks
$
```


Beispiel für IP sampling

- **„Visualisierung“ der Daten auf Modulebene**

```
$ vtl view -modules

VTune(TM) Performance Analyzer 2.0 for Linux* Update 1
Copyright (C) 2000-2004 Intel Corporation. All rights reserved.

Event Summary
Instructions Retired
   25996 = Samples collected due to this event
   2400000          = Sample after value used during collection
   62390400000      = Total events (samples*SAV)
Clockticks
   126400 = Samples collected due to this event
   2400000          = Sample after value used during collection
   303360000000     = Total events (samples*SAV)
```

1 Sample = 1ms bei 2,4 GHz

Beispiel für IP sampling

- Forts.

```
Module View (all values in decimal)

Module          Process
  Event                     Events% Samples Events     Module Path
-------------------------------------------------------------------
pic.ia32        pic.ia32
  Instructions Retired     99.81%  25947  62272800000 .../pic.ia32
  Clockticks               58.88%  74422 178612800000
vmlinux         Pid 0x0
  Instructions Retired      0.04%     10     24000000 vmlinux
  Clockticks               40.62% 51338  123211200000
...
```

Trotz leerer Maschine nur 58% der Clockticks im Userprogramm!

Beispiel für IP sampling

- „Hotspot View“ eröffnet Blick in ein Modul:

```
$ vtl view -hf -mn pic.ia32

Function        Module          Size
  Event                       Events% Samples Events [...]Full name
-------------------------------------------------------------------
MAIN            pic.ia32        0x6cc4
  Instructions Retired       58.72%   15264  36633600000 MAIN(void)
  Clockticks                 41.42%   52349 125637600000
ran1_           pic.ia32        0x248
  Instructions Retired       14.92%    3878  9307200000  ran1_
  Clockticks                  5.15%    6515 15636000000
mvteil_         pic.ia32        0x264
  Instructions Retired        9.68%    2516  6038400000  mvteil_
  Clockticks                  5.01%    6333 15199200000
```

Trotz „module view“ werden Anteile auf die gesamte Activity bezogen!

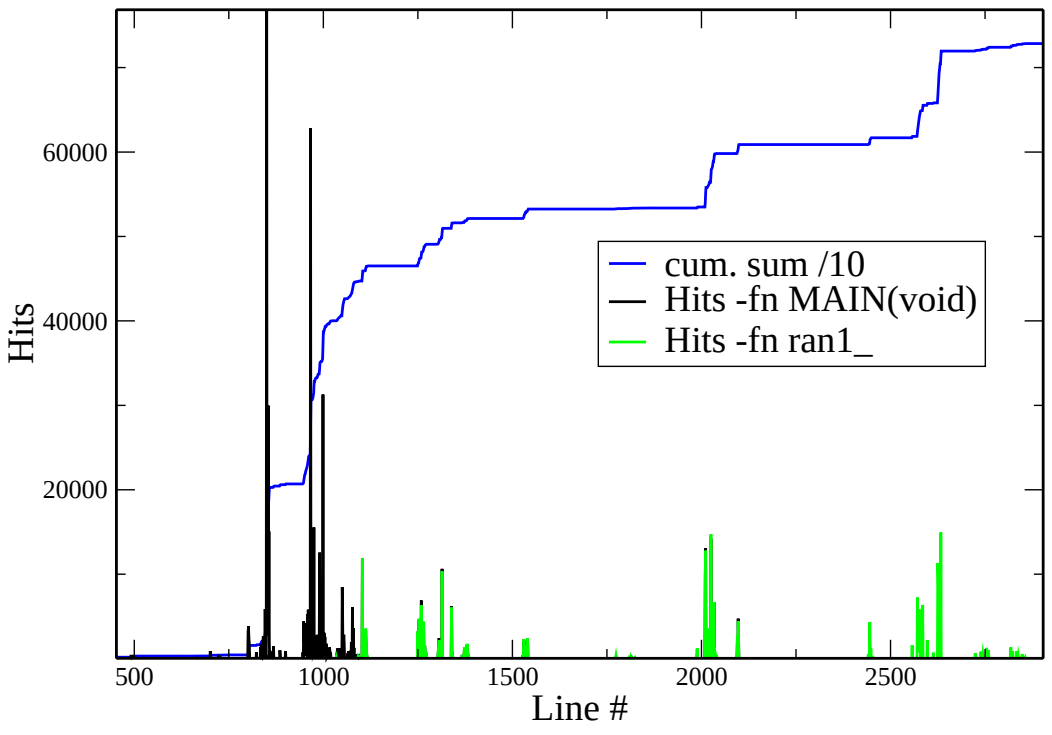
Beispiel für IP sampling

- „Source View“ listet Samples pro Codezeile in der Umgebung *einer bestimmten Funktion:*

```
$ vtl view -code -mn pic.ia32 -fn MAIN\(void\)

[...]
Legend
Ev1       =       Instructions Retired samples
Ev2       =       Clockticks samples
Address Line Number Ev1  Ev2 Source
0x1e4c  1           0    0
0x1dd4  2           0    0          PROGRAM  Plasma
[...]
0x3e4b  800         0    0            DO 200 I=1,N1
0x3e77  801         36   321             F(1,I) = 0.0
0x3e7f  802         33   368             F(2,I) = 0.0
0x3e87  803         28   256             F(3,I) = 0.0
0x3e8f  804         24   176  200      CONTINUE
```

- Option “`-sea pom`” listet Hits als % der Modul-Hits



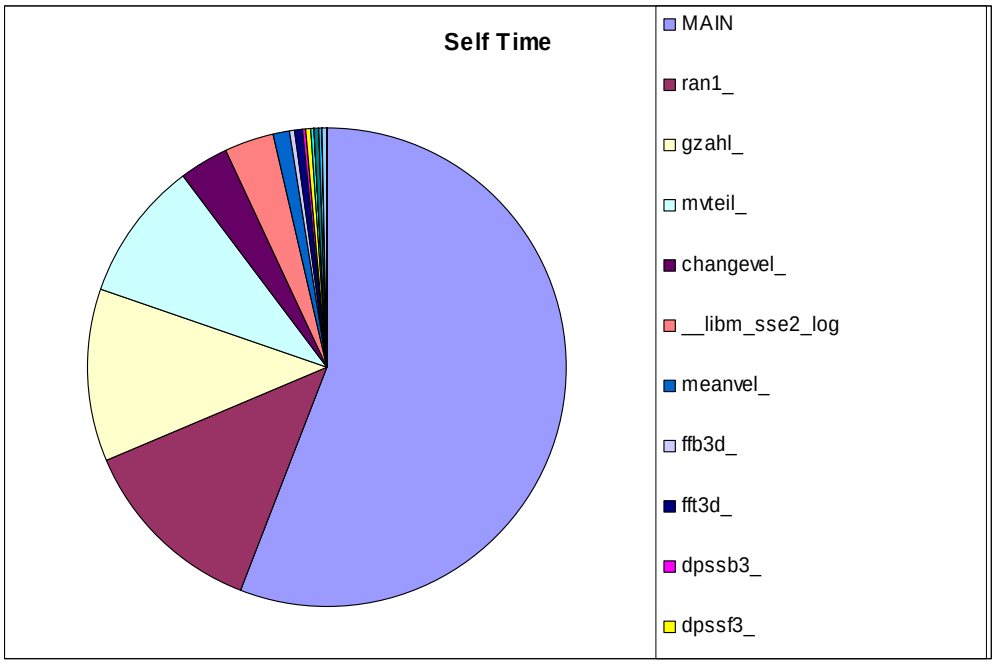
Callgraph Profiling

- Caller/Callee-Profile mit Ausgabe im CSV-Format
  - „Module of interest“ ist anzugeben
    - bei Sampling nicht sinnvoll, da dort immer das komplette System geprofiled wird
  - vtl instrumentiert das Executable und ev. Libraries vor dem Start

```
$ vtl activity -c callgraph \
  -app ./pic.ia32 \
  -moi pic.ia32 \
  run
[...]
$ vtl view -functions > pic_cg.csv
```

  - .csv in Excel laden und visualisieren (s. nächste Folie)
  - Informationen: Total time, self time, callers, callees, # of calls
  - Weitere Optionen filtern die Daten (`-calls`, `-critical-path` etc.)

Excel als Visualisierungstool



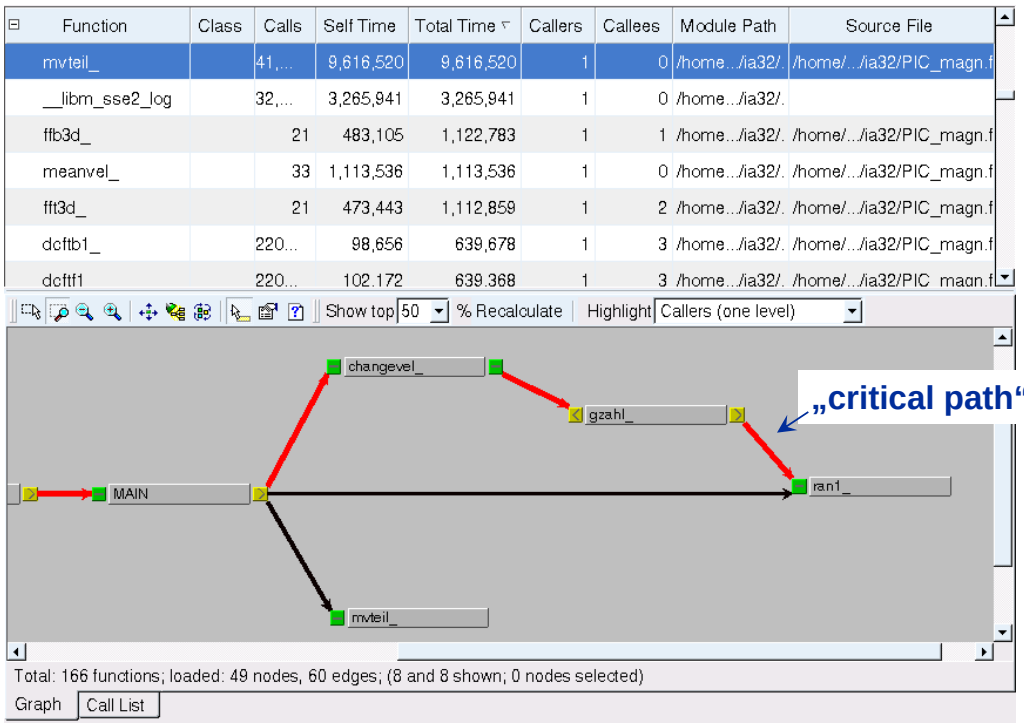
Callgraph Profiling

- GUI "cgviewer" ist seit VTune 2.0 Teil der Distribution
  - grafische Auswertung von callgraph-Daten
  - ahnlich cvperf unter IRIX, aber weniger mächtig
- Verwendung:

```
$ vtl view -gui
```

- Anzeige von
  - allen Subroutinen mit den üblichen Daten (total, self time etc.)
  - grafischem Calltree mit diversen Filterfunktionen
    - z.B. die 10 Funktionen mit größter self time etc.
  - "call list" mit Anzeige von Callern und Callees einer bestimmten Funktion
- Ähnliche Funktionalitäten sind in der Windows-Version von VTune integriert

Callgraph Viewer (GUI) unter Linux



Hardware Performance Counter

- **Ausgabe aller verwendbaren Sampling-Metriken:**

```
$ vtl query -c sampling

[... Optionen bei -c sampling ...]
[-ec | -event-config en|event-name= '<event name1>'
        [:sa|sample-after=<sample after value>],
                    en|'<event-name2>'...]
[...]
The supported CPU Events for this platform are as
below:

128-bit MMX(TM) Instructions Retired
1st Level Cache Load Misses Retired
2nd Level Cache Load Misses Retired
2nd Level Cache Read Misses
[...]
Non-Halted Clockticks
[...]
```


Hardware Performance Counter

- **Default-Optionen für eine Activity anzeigen:**

```
$ vtl query -a [activity]

[...]
Name: Activity3
Duration: 0 sec
Start with data collection paused: no

Application/Module Profiles:
  1. script
    Application to Launch:
      Filename: /home/cluster32/unrz/unrz55/GZ/pic/ia32/script
      System on which to launch application: <localhost>

  2. Modules of Interest
    Modules of Interest:
      1. pic.ia32
```

Default: so lange wie Applikation läuft

Folge von
-app ./script -moi pic.ia32

Hardware Performance Counter

- **Forts.**

```
Data Collectors:
  1. sampling
    Collector type: sampling
    Master: no
    Enabled: yes
    Associated with the following Application/Module
Profiles:
    1. script
    2. Modules of Interest
  Options:
    -cal no
    [...]
    -ec en=Clockticks:sa=2400000,en=Instructions \
Retired:sa=2400000
```

Kalibrierung (autom. Wahl des sample-after-value)

Eingestellte Metriken

Hardware Performance Counter

- **Beispiel: Messung der „Non-Halted Clockticks“ und der „64k Aliasing Conflicts”**
  - **was ist der richtige Wert für sample-after (sa) für jeden Counter?**
  - **1. Möglichkeit: "Schuss ins Blaue"**

```
$ vtl activity -c sampling -o \
  "-ec en='Non-Halted Clockticks':sa=2400000 \
      en='64k Aliasing Conflicts':sa=24000 \
      -cal no" -app ./pic.ia32 run
```

  - **2. Möglichkeit: VTune kann durch Kalibrierungsläufe vernünftige sa-Werte selbst bestimmen:**

```
$ vtl activity -d 100 -c sampling -o \
  "-ec en='Non-Halted Clockticks' \
      en='64k Aliasing Conflicts' \
      -cal yes" -app ./pic.ia32 run
```

Dabei ist die Angabe einer Duration (-d) Pflicht (Grund unklar)

Hardware Performance Counter

- **Module view zeigt jetzt keine seltsamen Hits für den Kernel mehr**
  - **`Clockticks` laufen weiter, auch wenn der Kernel schläft, `Non-Halted Clockticks` nicht**

```
$ vtl view -modules
Event Summary
64k Aliasing Conflicts
   158125 = Samples collected due to this event
   11475          = Sample after value used during collection
   1814484375     = Total events (samples*SAV)
Non-Halted Clockticks
   154953 = Samples collected due to this event
   1167610        = Sample after value used during collection
   180924672330   = Total events (samples*SAV)
```

via Kalibrierung durch VTune selbst bestimmt (zusätzlicher Lauf nötig)

Hardware Performance Counter

- **Forts.**

```
Module          Process
  Event                    Events% Samples Events       Module Path
-------------------------------------------------------------------
pic.ia32        pic.ia32
  64k Aliasing Conflicts   99.48%  157296   1804971600 …/pic.ia32
  Non-Halted Clockticks    99.07%  153509 179238643490
[...]
```

- **Source View zeigt dann Spalten (`EvN`) für alle ausgewählten Hardware Counter**
- **Bei Angabe von mehr Hardware-Countern als gleichzeitig gezählt werden können (max. 2) erfolgen automatisch mehrere Kalibrierungs- und Messläufe**

Hardware Performance Counter

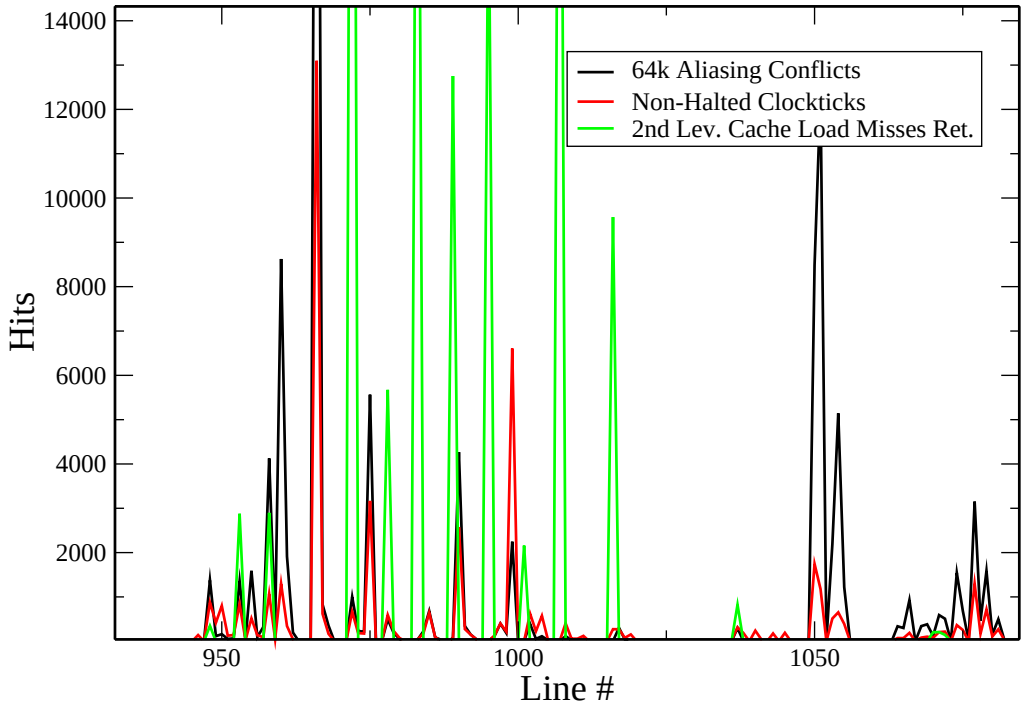
- Non-Halted Clockticks, 64k Aliasing Conflicts, 2nd Level Cache Load Misses Retired:

```
$ vtl activity -d 100 -c sampling -o
    "-ec en='Non-Halted Clockticks'  \
         en='64k Aliasing Conflicts' \
         en='2nd Level Cache Load Misses Retired' \
         -cal yes" -app ./pic.ia32
```

- Insgesamt 4 Läufe (2x Kalibrierung, 2x Messung)
- Modules und Hotspot View zeigt leider nicht alle Counter beisammen an, sondern in Zweiergruppen von

Hardware Performance Counter

```
Tue Jul 13 22:41:39 2004 storm (Run 0) The Sampling Collector is
calibrating its collection parameters for the following event
(s): 64k Aliasing Conflicts, Non-Halted Clockticks.
[...1. Lauf...]
Tue Jul 13 22:42:59 2004 storm (Run 0) The Sampling Collector is
collecting samples based on the following event(s): 64k Aliasing
Conflicts, Non-Halted Clockticks.
[...2. Lauf...]
Tue Jul 13 22:44:20 2004 storm (Run 0) Sampling data was
successfully collected.
Tue Jul 13 22:44:22 2004 storm (Run 1) The Sampling Collector is
calibrating its collection parameters for the following
event(s): 2nd Level Cache Load Misses Retired.
[...3. Lauf...]
Tue Jul 13 22:45:42 2004 storm (Run 1) The Sampling Collector is
collecting samples based on the following event(s): 2nd Level
Cache Load Misses Retired.
[...4. Lauf...]
Tue Jul 13 22:47:02 2004 storm (Run 1) Sampling data was
successfully collected.
The Activity has finished running.
```



Abschließendes

- **Sampling für das komplette System (Beispiel Flops für Itanium 2):**

```
$ vtl activity -d 10 -c sampling \
  -o "-ec en=FP_OPS_RETIRED -cal yes" run
```

- **Verschiedene Optionen beim Source view**
  - **Mischung mit Assemblercode in verschiedener Sortierung möglich**
- **ActivityController**
  - **kann benutzt werden, um eine laufende Activity zeitweilig anzuhalten oder zu stoppen**
  - **separates Programm**
- **Einpacken des Projekts zur Visualisierung unter Windows:**
  `$ vtl pack filename`

Abschließendes

- Sampling von MPI-Programmen auf einer Maschine:

```
$ vtl activity -c sampling -o \
  "-ec en='Non-Halted Clockticks':sa=2400000 -cal no" \
  -app /opt/mpich-1.2.4/bin/mpirun,\
       "-machinefile ./machines -np 3 ./a.out“ run
[...]
$ vtl view –processes
[...]
$ vtl view –hf –mn a.out –pid 12345
[...]
$ vtl view -code -mn ./a.out -fn shade –pid 12345
[...]
```

- Mit OpenMP funktioniert das aber nicht (Threads erscheinen nicht getrennt im Process View)

Abschließendes

- Zusammenfassung der Erfahrungen mit VTune 2 for Linux
  - Sehr nützliches Tool, extrem viele Optionen, nichts für Anfänger
  - Mangelhafte Unterstützung paralleler Programme
  - Gelegentliche Hänger
    - Restart aller VTune-Konponenten bzw. Reboot behebt das
  - Wünschenswert
    - Counter Multiplexing
    - Bessere Interoperabilität mit Windows-Version (für volle Funktionalität ist i.W. ein Samba-Zugriff auf die Linux-Maschine notwendig)
  - Sicherheitsproblem beim Remote Sampling (Port 50000)
  - VTune for Windows ist viel weiter
    - Bedienung, Hilfsfunktionen, Tuning Tips
  - VTune for Linux 3.0 (ende Q3/04) soll volle GUI haben

OProfile

- http://oprofile.sourceforge.net/
  - Work in progress, oft neue (inkompatible) Releases
  - Guter Support über Mailingliste
- Systemweites Profiling-Tool (Sampler) unter GPL
  - Zeit- oder Eventbasiertes Sampling
  - Seit Version 0.8 (akt.) experimentelles Callstack Sampling
  - Filtermöglichkeiten (Prozesse, CPUs)
  - Source view mit Event Counts, kein Multiplexing
  - Multi-Platform (IA32/64, Alpha, Power, x86-64)
- Kernelmodul plus Sampling Daemon plus Tools
- Größter Nachteil: Tools müssen als root ausgeführt werden (setuid root genügt nicht)
- Am RRZE
  - Erfolgreiche Tests mit Xeon- Itanium2- und Opteron-CPUs
  - Inkompatibel mit VTune
  - Aus jetziger Sicht ist VTune der Vorzug zu geben