Profiling Rexx with **bpf** and **perf**

René Vincent Jansen, Performance Architect, Department of Finance - Customs
The Netherlands
November 2021

BPF

- BPF (Berkely Packet Filter) is part of the kernel
  - Available in every modern Linux kernel
  - Perf and bpftrace are tools for working with bpf

- ‘Kernel VM’ programs, can be made with bcc (hard), bpftrace (easier)
- Perf is a command that is useful for most profiling/sampling/tracing actions

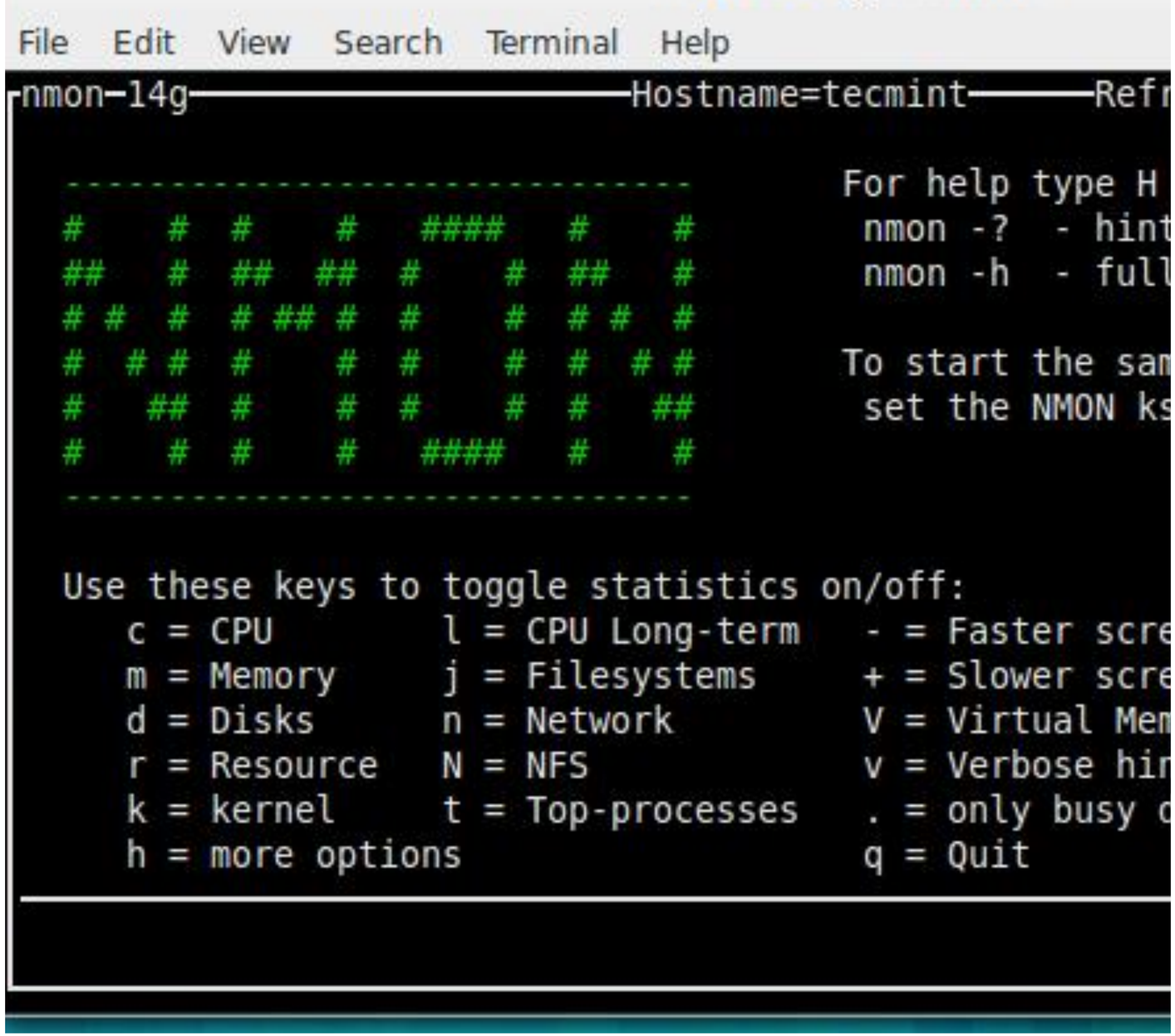
Top

Every Linux has this

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1065 MYSQL     20   0 2489680 420912  36020 S   0,3   2,6  26:51.46 MYSQLD
    1 ROOT      20   0  169940  11992   8432 S   0,0   0,1   0:40.68 SYSTEMD
    2 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.12 KTHREADD
    3 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 RCU_GP
    4 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 RCU_PAR_GP
    6 ROOT       0 -20       0      0      0 I   0,0   0,0   0:02.98 KWORKER/0:0H-KBLOCKD
    9 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 MM_PERCPU_WQ
   10 ROOT      20   0       0      0      0 S   0,0   0,0   1:41.77 KSOFTIRQD/0
   11 ROOT      20   0       0      0      0 I   0,0   0,0  10:50.65 RCU_SCHED
   12 ROOT      RT   0       0      0      0 S   0,0   0,0   0:02.22 MIGRATION/0
   13 ROOT     -51   0       0      0      0 S   0,0   0,0   0:00.00 IDLE_INJECT/0
   14 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 CPUHP/0
   15 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 CPUHP/1
   16 ROOT     -51   0       0      0      0 S   0,0   0,0   0:00.00 IDLE_INJECT/1
   17 ROOT      RT   0       0      0      0 S   0,0   0,0   0:02.34 MIGRATION/1
   18 ROOT      20   0       0      0      0 S   0,0   0,0   0:34.30 KSOFTIRQD/1
   20 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 KWORKER/1:0H-EVENTS_HIGHPRI
   21 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 CPUHP/2
   22 ROOT     -51   0       0      0      0 S   0,0   0,0   0:00.00 IDLE_INJECT/2
   23 ROOT      RT   0       0      0      0 S   0,0   0,0   0:02.20 MIGRATION/2
   24 ROOT      20   0       0      0      0 S   0,0   0,0   0:46.30 KSOFTIRQD/2
   26 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 KWORKER/2:0H-KBLOCKD
   27 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 CPUHP/3
   28 ROOT     -51   0       0      0      0 S   0,0   0,0   0:00.00 IDLE_INJECT/3
   29 ROOT      RT   0       0      0      0 S   0,0   0,0   0:02.47 MIGRATION/3
   30 ROOT      20   0       0      0      0 S   0,0   0,0   1:03.03 KSOFTIRQD/3
   32 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 KWORKER/3:0H-KBLOCKD
   33 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 KDEVTMPFS
   34 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 NETNS
   35 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 RCU_TASKS_KTHRE
   36 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 KAUDITD
   37 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.44 KHUNGTASKD
   38 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 OOM_REAPER
   39 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 WRITEBACK
   40 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 KCOMPACTD0
   41 ROOT      25   5       0      0      0 S   0,0   0,0   0:00.00 KSMD
   42 ROOT      39  19       0      0      0 S   0,0   0,0   0:02.08 KHUGEPAGED
   89 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 KINTEGRITYD
   90 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 KBLOCKD
   91 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 BLKCG_PUNT_BIO
   93 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 TPM_DEV_WQ
   94 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 ATA_SFF
   95 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 MD
   96 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 EDAC-POLLER
   97 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 DEVFREQ_WQ
   98 ROOT      RT   0       0      0      0 S   0,0   0,0   0:00.00 WATCHDOGD
  101 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 KSWAPD0
  102 ROOT      20   0       0      0      0 S   0,0   0,0   0:00.00 ECRYPTFS-KTHREA
  104 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 KTHROTLD
  105 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 ACPI_THERMAL_PM
  106 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 VFIO-IRQFD-CLEA
  108 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 IPV6_ADDRCONF
  118 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 KSTRP
  121 ROOT       0 -20       0      0      0 I   0,0   0,0   0:01.74 KWORKER/U9:0-HCI0
  137 ROOT       0 -20       0      0      0 I   0,0   0,0   0:00.00 CHARGER_MANAGER
```

Look at the system

- Top
- NMON
- Perf top



NMON

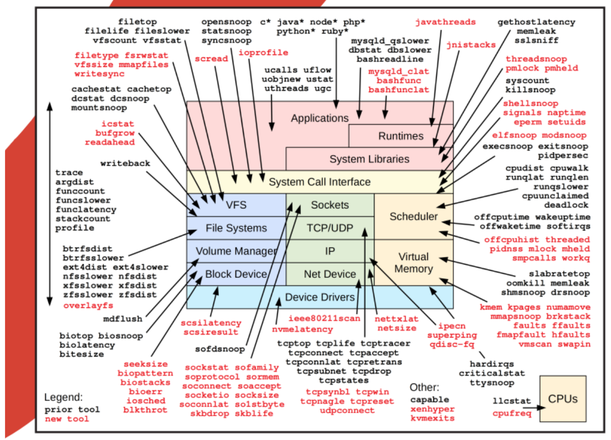
This needs installing

It will also give you a performance overview on the process level

BPF Performance Tools

Linux System and Application Observability

Brendan Gregg



Foreword by Alexei Starovoitov, creator of the new BPF

BPF and Perf

BPF (“Berkeley Packet Filter”) and perf-events are part of the Linux kernel. There is a ‘**kernel VM**’ that enables one to write small programs to be executed by the kernel in a controlled way.

perf, **top** and **iostat** are, nowadays, built on top of that. Perf and bpftrace need to be installed.

bcc and **bpftrace** are ways to make these kernel vm programs, the latter being modelled on **awk**.

Profile -why

- Good question!
- We want to know where the time is spent. Process-level is not enough to give developers clues on what to speed up.
- Not every program is optimally designed and implemented
- I will present a short series of (5) examples to give you an idea how much structures and algorithms can influence the performance of simple tasks
- [*these are a bit contrived (running 10000 times) to show some mechanisms clearly.*]

№ 1

Perf stat

We want to know more than the ‘time’

Perf stat

```
➜  bpfsprobes git:(master) X perf stat ./dhrystone
Dhrystone(1.1) time for 50000000 passes = 4
This machine benchmarks at 11029411 dhrystones/second

 Performance counter stats for './dhrystone':

          2.743,86 msec task-clock                #    1,000 CPUs utilized
                 9      context-switches          #    0,003 K/sec
                 1      cpu-migrations            #    0,000 K/sec
                54      page-faults               #    0,020 K/sec
    10.893.802.753      cycles                    #    3,970 GHz
    29.661.436.752      instructions              #    2,72  insn per cycle
     3.752.113.617      branches                  # 1367,460 M/sec
            71.092      branch-misses             #    0,00% of all branches

       2,744234624 seconds time elapsed

       2,744167000 seconds user
       0,000000000 seconds sys
```

Demo

```
Gen|1|1| In the beginning God created the heaven and the earth.~
Gen|1|2| And the earth was without form, and void; and darkness was upon the face of the deep. And the Spirit of God moved upon
 the face of the waters.~
Gen|1|3| And God said, Let there be light: and there was light.~
Gen|1|4| And God saw the light, that it was good: and God divided the light from the darkness.~
Gen|1|5| And God called the light Day, and the darkness he called Night. And the evening and the morning were the first day.~
Gen|1|6| And God said, Let there be a firmament in the midst of the waters, and let it divide the waters from the waters.~
Gen|1|7| And God made the firmament, and divided the waters which were under the firmament from the waters which were above the
 firmament: and it was so.~
Gen|1|8| And God called the firmament Heaven. And the evening and the morning were the second day.~
Gen|1|9| And God said, Let the waters under the heaven be gathered together unto one place, and let the dry land appear: and it
 was so.~
Gen|1|10| And God called the dry land Earth; and the gathering together of the waters called he Seas: and God saw that it was g
ood.~
Gen|1|11| And God said, Let the earth bring forth grass, the herb yielding seed, and the fruit tree yielding fruit after his ki
nd, whose seed is in itself, upon the earth: and it was so.~
Gen|1|12| And the earth brought forth grass, and herb yielding seed after his kind, and the tree yielding fruit, whose seed was
 in itself, after his kind: and God saw that it was good.~
Gen|1|13| And the evening and the morning were the third day.~
Gen|1|14| And God said, Let there be lights in the firmament of the heaven to divide the day from the night; and let them be fo
r signs, and for seasons, and for days, and years:~
Gen|1|15| And let them be for lights in the firmament of the heaven to give light upon the earth: and it was so.~
Gen|1|16| And God made two great lights; the greater light to rule the day, and the lesser light to rule the night: he made the
 stars also.~
Gen|1|17| And God set them in the firmament of the heaven to give light upon the earth,~
Gen|1|18| And to rule over the day and over the night, and to divide the light from the darkness: and God saw that it was good.
~
Gen|1|19| And the evening and the morning were the fourth day.~
Gen|1|20| And God said, Let the waters bring forth abundantly the moving creature that hath life, and fowl that may fly above t
he earth in the open firmament of heaven.~
Gen|1|21| And God created great whales, and every living creature that moveth, which the waters brought forth abundantly, after
 their kind, and every winged fowl after his kind: and God saw that it was good.~
Gen|1|22| And God blessed them, saying, Be fruitful, and multiply, and fill the waters in the seas, and let fowl multiply in th
e earth.~
Gen|1|23| And the evening and the morning were the fifth day.~
Gen|1|24| And God said, Let the earth bring forth the living creature after his kind, cattle, and creeping thing, and beast of
the earth after his kind: and it was so.~
Gen|1|25| And God made the beast of the earth after his kind, and cattle after their kind, and every thing that creepeth upon t
he earth after his kind: and God saw that it was good.~
Gen|1|26| And God said, Let us make man in our image, after our likeness: and let them have dominion over the fish of the sea,
and over the fowl of the air, and over the cattle, and over all the earth, and over every creeping thing that creepeth upon the
 earth.~
Gen|1|27| So God created man in his own image, in the image of God created he him; male and female created he them.~
Gen|1|28| And God blessed them, and God said unto them, Be fruitful, and multiply, and replenish the earth, and subdue it: and
have dominion over the fish of the sea, and over the fowl of the air, and over every living thing that moveth upon the earth.~
Gen|1|29| And God said, Behold, I have given you every herb bearing seed, which is upon the face of all the earth, and every tr
ee, in the which is the fruit of a tree yielding seed; to you it shall be for meat.~
Gen|1|30| And to every beast of the earth, and to every fowl of the air, and to every thing that creepeth upon the earth, where
in there is life, I have given every green herb for meat: and it was so.~
Gen|1|31| And God saw every thing that he had made, and, behold, it was very good. And the evening and the morning were the six
th day.~
Gen|2|1| Thus the heavens and the earth were finished, and all the host of them.~
Gen|2|2| And on the seventh day God ended his work which he had made; and he rested on the seventh day from all his work which
he had made.~
Gen|2|3| And God blessed the seventh day, and sanctified it: because that in it he had rested from all his work which God creat
ed and made.
Ge
```

We have a text

KJV= King James Version of the Bible

4521345 bytes (4.4 MB)

We try for 10000 times to find the last verse of Revelations

The first attempts are not particularly bright

We have a Class

This is a container. We make an instance of it for every line from the text, and use PARSE to fill it.

Only the later programs in the series are using it

```
CLASS ChapterAndVerse

PROPERTIES PUBLIC

book
chapter
verse
textLine

  METHOD ChapterAndVerse(book_,chapter_,verse_,textLine_)
    book = book_
    chapter = chapter_
    verse = verse_
    textLine = textLine_

  METHOD toString() RETURNS String
    RETURN book chapter verse textLine
```

First try: we loop with I/O

```
CLASS Read_Text_Loop

  METHOD Read_Text_Loop()

  METHOD main(args=String[]) STATIC
    r=Read_Text_Loop()
    textLine=''
    LOOP FOR 10000
      source = BufferedReader(FileReader('./data/kjvdat.txt'))
      LOOP FOREVER
        lastLine=textLine
        textLine = source.readLine
        IF textLine = null THEN LEAVE
      END
    END
    SAY lastLine
```

```
➜  bpfsprobes git:(master) perf stat java -XX:+PreserveFramePointer  Read_Text_Loop
Rev|22|21| The grace of our Lord Jesus Christ be with you all. Amen.~

 Performance counter stats for 'java -XX:+PreserveFramePointer Read_Text_Loop':

         54.821,02 msec task-clock                #    1,009 CPUs utilized
            33.517      context-switches          #    0,611 K/sec
             4.825      cpu-migrations            #    0,088 K/sec
            74.516      page-faults               #    0,001 M/sec
   214.181.041.636      cycles                    #    3,907 GHz
   634.317.494.047      instructions              #    2,96  insn per cycle
   141.624.924.841      branches                  # 2583,406 M/sec
     1.353.808.195      branch-misses             #    0,96% of all branches

      54,334276354 seconds time elapsed

      49,806402000 seconds user
       5,269838000 seconds sys
```

Second Program: another sort of Read loop

Here we are assigning the content to an instance of ChapterAndVerse

Apart from the construct, does more or less the same. We read only once. Notation-wise is the base for the examples; it is much shorter; also this is the performance baseline.

```
CLASS Read_Text_Oneline

PROPERTIES INHERITABLE
aL = ArrayList()

  METHOD Read_Text_Oneline()
    RexxIO().File('./data/kjvdat.txt').forEachline(THIS.docid())

  METHOD main(args=String[]) STATIC
    r=Read_Text_Oneline()

  CLASS Read_Text_Oneline.docid DEPENDENT IMPLEMENTS LineHandler
  METHOD handle(in)

    PARSE in book '|' chapter '|' verse '|' textLine
    a = ChapterAndVerse(book,chapter,verse,textLine)
    PARENT.aL.add(a)
```

```
➜  bpfsprobes git:(master) perf stat java -XX:+PreserveFramePointer  Read_Text_Oneline

 Performance counter stats for 'java -XX:+PreserveFramePointer Read_Text_Oneline':

            312,48 msec task-clock                #    2,012 CPUs utilized
               369      context-switches          #    0,001 M/sec
                52      cpu-migrations            #    0,166 K/sec
            15.988      page-faults               #    0,051 M/sec
     1.195.562.392      cycles                    #    3,826 GHz
     1.440.566.467      instructions              #    1,20  insn per cycle
       264.599.270      branches                  #  846,775 M/sec
         7.326.681      branch-misses             #    2,77% of all branches

       0,155274554 seconds time elapsed

       0,289902000 seconds user
       0,024498000 seconds sys
```

Avoiding the I/O

We have an ArrayList

We add instances of ChapterAndVerse

Now we read only once and loop 10000 times through this, until we are at the end.

We are not looking or comparing yet.

```
CLASS Read_Text_Oneline1

PROPERTIES INHERITABLE
aL = ArrayList()

  METHOD Read_Text_Oneline1()
    RexxIO().File('./data/kjvdat.txt').forEachline(THIS.docid())

  METHOD main(args=String[]) STATIC
    r=Read_Text_Oneline1()
    LOOP FOR 10000
      LOOP i=0 TO r.aL.SIZE()-1
        a = ChapterAndVerse r.aL.GET(i)
      END
    END
    SAY a

  CLASS Read_Text_Oneline1.docid DEPENDENT IMPLEMENTS LineHandler
  METHOD handle(in)

    PARSE in book '|' chapter '|' verse '|' textLine
    a = ChapterAndVerse(book,chapter,verse,textLine)
    PARENT.aL.add(a)
```

```
➜  bpfsprobes git:(master) perf stat java -XX:+PreserveFramePointer  Read_Text_Oneline1
Rev 22 21  The grace of our Lord Jesus Christ be with you all. Amen.~

 Performance counter stats for 'java -XX:+PreserveFramePointer Read_Text_Oneline1':

         16.805,86 msec task-clock                #    1,025 CPUs utilized
             6.341      context-switches          #    0,377 K/sec
               908      cpu-migrations            #    0,054 K/sec
            93.276      page-faults               #    0,006 M/sec
    66.487.566.474      cycles                    #    3,956 GHz
   200.339.033.204      instructions              #    3,01  insn per cycle
    36.499.595.263      branches                  # 2171,837 M/sec
        48.721.562      branch-misses             #    0,13% of all branches

      16,398101426 seconds time elapsed

      16,710133000 seconds user
       0,136311000 seconds sys
```

Now look for the right verse in the ArrayList

We again loop through the ArrayList that we filled once

But now we search for the book of Revelations 22:21

Which is the last line we added to the Arraylist

```
CLASS Read_Text_Oneline2

PROPERTIES INHERITABLE
aL = ArrayList()

  METHOD Read_Text_Oneline2()
    RexxIO().File('./data/kjvdat.txt').forEachline(THIS.docid())

  METHOD main(args=String[]) STATIC
    r=Read_Text_Oneline2()
    LOOP FOR 10000
      LOOP i=0 TO r.aL.size()-1
        a = ChapterAndVerse r.aL.get(i)
        IF a.book='Rev' THEN
          IF a.chapter='22' THEN
            IF a.verse='21' THEN LEAVE
      END
    END
    SAY a

  CLASS Read_Text_Oneline2.docid DEPENDENT IMPLEMENTS LineHandler
  METHOD handle(in)

    PARSE in book '|' chapter '|' verse '|' textLine
    a = ChapterAndVerse(book,chapter,verse,textLine)
    PARENT.aL.add(a)
```

```
➜  bpfsprobes git:(master) perf stat java -XX:+PreserveFramePointer  Read_Text_Oneline2
Rev 22 21  The grace of our Lord Jesus Christ be with you all. Amen.~

 Performance counter stats for 'java -XX:+PreserveFramePointer Read_Text_Oneline2':

          27.941,67 msec task-clock                #    1,013 CPUs utilized
              6.797      context-switches          #    0,243 K/sec
              1.083      cpu-migrations            #    0,039 K/sec
             92.130      page-faults               #    0,003 M/sec
    110.927.691.194      cycles                    #    3,970 GHz
    240.753.224.323      instructions              #    2,17  insn per cycle
     46.301.682.611      branches                  # 1657,084 M/sec
         49.056.142      branch-misses             #    0,11% of all branches

       27,572957882 seconds time elapsed

       27,810831000 seconds user
        0,176500000 seconds sys
```

What happens when we want an earlier line

So so the time spent into searching in an Array is dependent of the position of the target string.

```
CLASS Read_Text_Oneline3

PROPERTIES INHERITABLE
aL = ArrayList()

  METHOD Read_Text_Oneline3()
    RexxIO().File('./data/kjvdat.txt').forEachline(THIS.docid())

  METHOD main(args=String[]) STATIC
    r=Read_Text_Oneline3()
    LOOP FOR 10000
      LOOP i=0 TO r.aL.size()-1
        a = ChapterAndVerse r.aL.get(i)
        IF a.book='Exo' THEN
          IF a.chapter='20' THEN
            IF a.verse='17' THEN LEAVE
      END
    END
    SAY a

  CLASS Read_Text_Oneline3.docid DEPENDENT IMPLEMENTS LineHandler
  METHOD handle(in)

    PARSE in book '|' chapter '|' verse '|' textLine
    a = ChapterAndVerse(book,chapter,verse,textLine)
    PARENT.aL.add(a)
```

```
→  bpfsprobes git:(master) perf stat java -XX:+PreserveFramePointer  Read_Text_Oneline3
Exo 20 17  Thou shalt not covet thy neighbour's house, thou shalt not covet thy neighbour's wife,                RVANT,
THING THAT IS THY NEIGHBOUR'S.~

 Performance counter stats for 'java -XX:+PreserveFramePointer Read_Text_Oneline3':

          1.740,40 msec task-clock                #    1,225 CPUs utilized
               923      context-switches          #    0,530 K/sec
               161      cpu-migrations            #    0,093 K/sec
            89.784      page-faults               #    0,052 M/sec
     6.829.614.161      cycles                    #    3,924 GHz
    17.993.925.496      instructions              #    2,63  insn per cycle
     3.395.732.302      branches                  # 1951,119 M/sec
        14.416.862      branch-misses             #    0,42% of all branches

       1,420480527 seconds time elapsed

       1,646809000 seconds user
       0,101404000 seconds sys
```

Let’s redo this with another structure, the TreeMap

A TreeMap is a keyed structure in which the keys and their values are stored in a sorted way.

This means looking up the key can be done by a binary search algorithm, that is convently hidden from us.

```
CLASS Read_Text_Oneline4

PROPERTIES INHERITABLE
tM = TreeMap()

  METHOD Read_Text_Oneline4()
    RexxIO().File('./data/kjvdat.txt').forEachline(THIS.docid())

  METHOD main(args=String[]) STATIC
    r=Read_Text_Oneline4()
    LOOP FOR 10000
      key='Rev|22|21'
      textLine = Rexx r.tM.get(key)
    END
    SAY textLine

  CLASS Read_Text_Oneline4.docid DEPENDENT IMPLEMENTS LineHandler
  METHOD handle(in)

    PARSE in book '|' chapter '|' verse '|' textLine
    a = ChapterAndVerse(book,chapter,verse,textLine)
    key=book'|'chapter'|'verse
    PARENT.tM.put(key,textLine)
```

```
➜  bpfsprobes git:(master) perf stat java -XX:+PreserveFramePointer  Read_Text_Oneline4
 The grace of our Lord Jesus Christ be with you all. Amen.~

 Performance counter stats for 'java -XX:+PreserveFramePointer Read_Text_Oneline4':

            612,42 msec task-clock                #    2,402 CPUs utilized
               422      context-switches          #    0,689 K/sec
                55      cpu-migrations            #    0,090 K/sec
            29.785      page-faults               #    0,049 M/sec
     2.354.285.302      cycles                    #    3,844 GHz
     2.807.428.519      instructions              #    1,19  insn per cycle
       537.083.134      branches                  #  876,987 M/sec
        12.404.891      branch-misses             #    2,31% of all branches

       0,254938162 seconds time elapsed

       0,566838000 seconds user
       0,048935000 seconds sys
```

Now try to find an earlier verse

Because of binary search, this is not quicker than looking for the last verse.

```
CLASS Read_Text_Oneline5

PROPERTIES INHERITABLE
tM = TreeMap()

  METHOD Read_Text_Oneline5()
    RexxIO().File('./data/kjvdat.txt').forEachline(THIS.docid())

  METHOD main(args=String[]) STATIC
    r=Read_Text_Oneline5()
    LOOP FOR 10000
      key='Exo|20|17'
      textLine = Rexx r.tM.get(key)
    END
    SAY textLine

  CLASS Read_Text_Oneline5.docid DEPENDENT IMPLEMENTS LineHandler
  METHOD handle(in)

    PARSE in book '|' chapter '|' verse '|' textLine
    a = ChapterAndVerse(book,chapter,verse,textLine)
    key=book'|'chapter'|'verse
    PARENT.tM.put(key,textLine)
```

```
➜  bpfsprobes git:(master) perf stat java -XX:+PreserveFramePointer  Read_Text_Oneline5
 Thou shalt not covet thy neighbour's house, thou shalt not covet thy neighbour's wife, nor his ma                  r his ma
thy neighbour's.~

 Performance counter stats for 'java -XX:+PreserveFramePointer Read_Text_Oneline5':

            617,96 msec task-clock                #    2,451 CPUs utilized
               413      context-switches          #    0,668 K/sec
                50      cpu-migrations            #    0,081 K/sec
            29.095      page-faults               #    0,047 M/sec
     2.381.035.553      cycles                    #    3,853 GHz
     2.768.066.794      instructions              #    1,16  insn per cycle
       527.846.156      branches                  #  854,169 M/sec
        12.321.448      branch-misses             #    2,33% of all branches

       0,252142547 seconds time elapsed

       0,535193000 seconds user
       0,085943000 seconds sys
```

№ 2

sudo perf top

Because sometimes you cannot run programs in isolation, and need a broad picture about what’s happening

```
        WHILE (IntLoc1 < IntLoc2)
        {
                IntLoc3 = 5 * IntLoc1 - IntLoc2;
                Proc7(IntLoc1, IntLoc2, &IntLoc3);
                ++IntLoc1;
        }
        Proc8(Array1Glob, Array2Glob, IntLoc1, IntLoc3);
        Proc1(PtrGlb);
        FOR (CharIndex = 'A'; CharIndex <= Char2Glob; ++CharIndex)
                IF (EnumLoc == Func1(CharIndex, 'C'))
                        Proc6(Ident1, &EnumLoc);
        IntLoc3 = IntLoc2 * IntLoc1;
        IntLoc2 = IntLoc3 / IntLoc1;
        IntLoc2 = 7 * (IntLoc3 - IntLoc2) - IntLoc1;
        Proc2(&IntLoc1);
    }
/*****************
-- Stop Timer --
*****************/
#IFDEF TIME
    BENCHTIME = TIME( (LONG *) 0) - STARTTIME - NULLTIME;
    PRINTF("Dhrystone(%s) time for %ld passes = %ld\n",
        Version,
        (LONG) LOOPS, BENCHTIME);
    PRINTF("This machine benchmarks at %ld dhrystones/second\n",
        ((LONG) LOOPS) / BENCHTIME);
#ENDIF
#IFDEF TIMES
    TIMES(&TMS);
    BENCHTIME = TMS.TMS_UTIME - STARTTIME - NULLTIME;
    PRINTF("Dhrystone(%s) time for %ld passes = %ld\n",
        Version,
        (LONG) LOOPS, BENCHTIME/HZ);
    PRINTF("This machine benchmarks at %ld dhrystones/second\n",
        ((LONG) LOOPS) * HZ / BENCHTIME);
#ENDIF
#IFDEF GETRUSAGE
    GETRUSAGE(RUSAGE_SELF, &ENDTIME);
    {
        DOUBLE T = (DOUBLE)(ENDTIME.RU_UTIME.TV_SEC
                            - STARTTIME.RU_UTIME.TV_SEC
                            - NULLTIME.TV_SEC)
                 + (DOUBLE)(ENDTIME.RU_UTIME.TV_USEC
                            - STARTTIME.RU_UTIME.TV_USEC
                            - NULLTIME.TV_USEC) * 1E-6;
        PRINTF("Dhrystone(%s) time for %ld passes = %.1f\n",
              Version,
              (LONG)LOOPS,
              T);
```

The Dhrystone benchmark

To generate some load for **perf top** and see where its time is spent.

The -Wno-imolicit-function-declaration is only for the M1 Mac because the source is so old and **clang** does not like it. There are no symbols or debug options selected.

```
CC -WNO-IMPLICIT-FUNCTION-DECLARATION DHRYSTONE.C -O DHRYSTONE
```

perf top of the functions within a process

We see exactly which functions in the program used the majority of the processor cycles. This gives us a handle on the optimising process.

```
Samples: 17K of event 'cycles', 4000 Hz, Event count (approx.): 6976335389 lost: 0/0 drop: 0/0
Overhead  Shared Object                     Symbol
  22,11%  dhrystone                         [.] Proc0
  16,16%  dhrystone                         [.] Proc1
  13,01%  dhrystone                         [.] Proc8
   7,04%  dhrystone                         [.] Func1
   6,76%  dhrystone                         [.] Proc7
   6,42%  dhrystone                         [.] Func2
   6,17%  libc-2.31.so                      [.] __strcmp_avx2
   6,11%  dhrystone                         [.] Proc3
   3,91%  dhrystone                         [.] Proc6
   2,56%  dhrystone                         [.] Proc2
   1,86%  dhrystone                         [.] Func3
   1,73%  dhrystone                         [.] Proc4
   0,76%  dhrystone                         [.] Proc5
   0,51%  dhrystone                         [.] 0x000000000000010c4
   0,09%  perf                              [.] __symbols__insert
   0,09%  perf                              [.] rb_next
   0,08%  [kernel]                          [k] psi_task_change
   0,07%  libslang.so.2.3.2                 [.] SLsmg_write_chars
   0,07%  perf                              [.] d_print_comp_inner
   0,06%  [kernel]                          [k] do_syscall_64
   0,06%  [kernel]                          [k] update_blocked_averages
   0,04%  [kernel]                          [k] menu_select
   0,04%  [kernel]                          [k] entry_SYSCALL_64
   0,04%  [kernel]                          [k] timerqueue_add
   0,04%  [kernel]                          [k] kallsyms_expand_symbol.constprop.0
   0,04%  [kernel]                          [k] __update_load_avg_cfs_rq
   0,03%  libc-2.31.so                      [.] _int_malloc
   0,03%  libslang.so.2.3.2                 [.] SLtt_smart_puts
   0,03%  perf                              [.] rust_demangle_callback
   0,03%  [kernel]                          [k] timekeeping_advance
   0,03%  snapd                             [.] 0x00000000004b1394
   0,03%  [kernel]                          [k] syscall_return_via_sysret
   0,03%  [kernel]                          [k] cpuidle_enter_state
   0,03%  [kernel]                          [k] _raw_spin_lock_irqsave
   0,03%  perf                              [.] dso__find_symbol
   0,03%  perf                              [.] hpp__sort_overhead
   0,03%  [kernel]                          [k] vsnprintf
   0,03%  [kernel]                          [k] module_get_kallsym
   0,03%  perf                              [.] rb_insert_color
   0,03%  perf                              [.] hists__findnew_entry
   0,03%  [kernel]                          [k] update_load_avg
   0,03%  perf                              [.] hist_entry__sort
   0,02%  [unknown]                         [.] 0000000000000000
   0,02%  [kernel]                          [k] __update_load_avg_se
   0,02%  [kernel]                          [k] __next_timer_interrupt
   0,02%  libc-2.31.so                      [.] __GI_____strtoull_l_internal
   0,02%  [kernel]                          [k] read_tsc
   0,02%  perf                              [.] d_print_comp
   0,02%  [kernel]                          [k] load_balance
   0,02%  libmozjs-68.so.68.6.0             [.] 0x00000000005a0984
   0,02%  [kernel]                          [k] format_decode
   0,02%  [kernel]                          [k] native_queued_spin_lock_slowpath
   0,02%  [kernel]                          [k] page_remove_rmap
   0,02%  [kernel]                          [k] switch_fpu_return
```

```
Samples: 83K of event 'cycles', 4000 Hz, Event count (approx.): 32391816297 lost: 0/0 drop: 0/0
Overhead  Shared Object                     Symbol
   4,66%  [kernel]                          [k] copy_user_enhanced_fast_string
   3,47%  [JIT] tid 102605                  [.] 0x00007f2e44c3a9d6
   3,15%  [JIT] tid 102605                  [.] 0x00007f2e44c3afe3
   2,54%  [JIT] tid 102605                  [.] 0x00007f2e44c3abb6
   2,51%  [JIT] tid 102605                  [.] 0x00007f2e44c3abde
   2,48%  [JIT] tid 102605                  [.] 0x00007f2e44c3ac40
   2,47%  [JIT] tid 102605                  [.] 0x00007f2e44c3abd5
   2,45%  [JIT] tid 102605                  [.] 0x00007f2e44c3ac14
   2,45%  [JIT] tid 102605                  [.] 0x00007f2e44c3abd0
   2,43%  [JIT] tid 102605                  [.] 0x00007f2e44c3abc0
   2,38%  [JIT] tid 102605                  [.] 0x00007f2e44c3abfa
   2,36%  [JIT] tid 102605                  [.] 0x00007f2e44c3ac38
   2,36%  [JIT] tid 102605                  [.] 0x00007f2e44c3ac1d
   2,36%  [JIT] tid 102605                  [.] 0x00007f2e44c3ac2a
   2,27%  [JIT] tid 102605                  [.] 0x00007f2e44c3ac00
   2,20%  [JIT] tid 102605                  [.] 0x00007f2e44c3abea
   2,10%  [JIT] tid 102605                  [.] 0x00007f2e44c416c4
   1,76%  [kernel]                          [k] do_syscall_64
   1,44%  [JIT] tid 102605                  [.] 0x00007f2e44c3e18f
   1,31%  libc-2.31.so                      [.] __memmove_avx_unaligned_erms
   0,99%  [JIT] tid 102605                  [.] 0x00007f2e44c3aef1
   0,91%  [JIT] tid 102605                  [.] 0x00007f2e44c415f4
   0,90%  [JIT] tid 102605                  [.] 0x00007f2e44c3e189
   0,88%  [kernel]                          [k] find_get_entry
   0,82%  [JIT] tid 102605                  [.] 0x00007f2e44c3ae6b
   0,80%  [JIT] tid 102605                  [.] 0x00007f2e44c415f9
   0,79%  [JIT] tid 102605                  [.] 0x00007f2e44c3ae70
   0,78%  [kernel]                          [k] entry_SYSCALL_64
   0,78%  [kernel]                          [k] syscall_return_via_sysret
   0,77%  [JIT] tid 102605                  [.] 0x00007f2e44c3ae7a
   0,72%  [JIT] tid 102605                  [.] 0x00007f2e44c416be
   0,61%  [JIT] tid 102605                  [.] 0x00007f2e44c3af18
   0,61%  [JIT] tid 102605                  [.] 0x00007f2e44c3ae7e
   0,60%  [JIT] tid 102605                  [.] 0x00007f2e44c3ae83
   0,58%  [JIT] tid 102605                  [.] 0x00007f2e44c33963
   0,58%  [JIT] tid 102605                  [.] 0x00007f2e44c3aedd
   0,46%  [JIT] tid 102605                  [.] 0x00007f2e44c3e207
   0,43%  [JIT] tid 102605                  [.] 0x00007f2e44c3aea4
   0,43%  [JIT] tid 102605                  [.] 0x00007f2e44c3ae8d
   0,43%  [JIT] tid 102605                  [.] 0x00007f2e44c3aee9
   0,43%  [JIT] tid 102605                  [.] 0x00007f2e44c3af4c
   0,36%  [JIT] tid 102605                  [.] 0x00007f2e44c3e1fa
   0,35%  [JIT] tid 102605                  [.] 0x00007f2e44c3ae74
   0,34%  [JIT] tid 102605                  [.] 0x00007f2e44c3acb8
   0,34%  [JIT] tid 102605                  [.] 0x00007f2e44c3acc9
   0,33%  [kernel]                          [k] generic_file_buffered_read
   0,33%  [JIT] tid 102605                  [.] 0x00007f2e44c338c0
   0,33%  [JIT] tid 102605                  [.] 0x00007f2e44c3acc4
```

Now let's run our first java program

The one with the loop

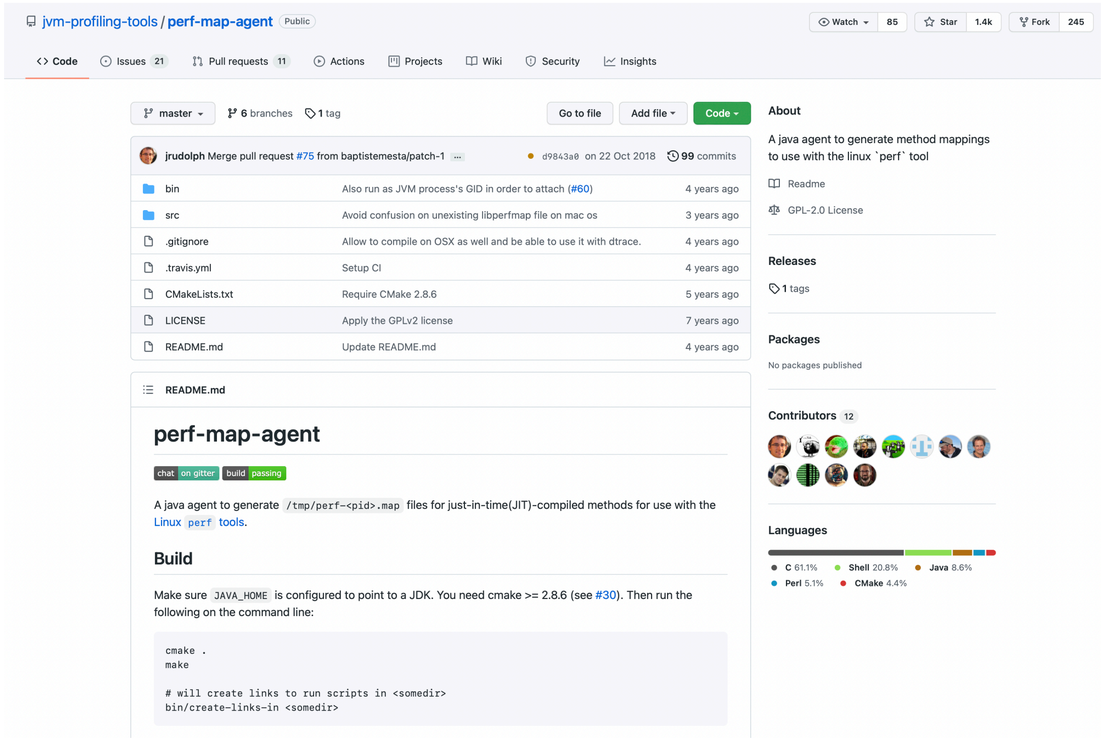
Very disappointing, isn't is?

What happens here? We have only addresses, no human-readable symbols.

This is caused by the JIT process, which does Just-In-Time compiling to native (instruction set architecture dependent) machine code.

Symbols to the rescue

- https://github.com/jvm-profiling-tools/perf-map-agent
- Set JAVA_HOME
- cmake .
- make



Create a map with java-perf-map

- java **-XX:+PreserveFramePointer** <your_class>
- sudo perf record -F 99 -p ‘pgrep java’ -g -- sleep 10
- ~/apps/perf-map-agent/bin/create-java-perf-map.sh ‘pgrep java’
- sudo perf script >out.perf
- But for now, we are going to look at **perf top**

Linux perf tools will expect symbols for code executed from unknown memory regions at **/tmp/perf-<pid>.map**. This allows runtimes that generate code on the fly to supply dynamic symbol mappings to be used with the perf suite of tools.

```
Samples: 51K of event 'cycles', 4000 Hz, Event count (approx.): 35793172624 lost: 0/0 drop: 0/0
Overhead  Shared Object             Symbol
  61,68%  [JIT] tid 102664          [.] Ljava/io/BufferedReader;::readLine
   8,63%  [JIT] tid 102664          [.] LRead_Text_Loop;::main
   5,16%  [JIT] tid 102664          [.] Lsun/nio/cs/UTF_8$Decoder;::decodeArrayLoop
   4,92%  [kernel]                  [k] copy_user_enhanced_fast_string
   2,13%  [JIT] tid 102664          [.] Lnetrexx/lang/Rexx;::<init>
   1,96%  [kernel]                  [k] do_syscall_64
   1,48%  libc-2.31.so              [.] __memmove_avx_unaligned_erms
   1,32%  [JIT] tid 102664          [.] Ljava/lang/AbstractStringBuilder;::appendChars
   0,86%  [kernel]                  [k] find_get_entry
   0,81%  [kernel]                  [k] syscall_return_via_sysret
   0,75%  [kernel]                  [k] entry_SYSCALL_64
   0,53%  [kernel]                  [k] generic_file_buffered_read
   0,41%  [JIT] tid 102664          [.] Ljava/io/InputStreamReader;::read
   0,32%  [JIT] tid 102664          [.] Ljava/io/FileInputStream;::readBytes
   0,30%  [JIT] tid 102664          [.] jbyte_disjoint_arraycopy
   0,25%  [JIT] tid 102664          [.] Ljava/lang/AbstractStringBuilder;::ensureCapacityInternal
   0,25%  perf                      [.] __symbols__insert
   0,22%  [kernel]                  [k] xas_load
   0,21%  [kernel]                  [k] __fget
   0,21%  perf                      [.] d_print_comp_inner
   0,16%  [kernel]                  [k] ext4_file_read_iter
   0,16%  perf                      [.] rb_next
   0,15%  [kernel]                  [k] copy_page_to_iter
   0,15%  [kernel]                  [k] common_file_perm
   0,15%  libc-2.31.so              [.] read
   0,14%  libjvm.so                 [.] 0x0000000000a9dc50
   0,09%  [kernel]                  [k] vfs_read
   0,09%  [kernel]                  [k] kallsyms_expand_symbol.constprop.0
   0,09%  [kernel]                  [k] module_get_kallsym
   0,08%  [kernel]                  [k] fsnotify
   0,08%  [kernel]                  [k] ksys_read
   0,08%  libjvm.so                 [.] 0x00000000008bc5ed
   0,08%  libjvm.so                 [.] 0x00000000008c4aa2
   0,08%  [kernel]                  [k] new_sync_read
   0,08%  perf                      [.] rust_demangle_callback
   0,08%  perf                      [.] d_print_comp
   0,08%  perf                      [.] dso__find_symbol
   0,08%  [kernel]                  [k] vsnprintf
   0,07%  [JIT] tid 102664          [.] Ljava/util/Arrays;::copyOfRange
   0,07%  libc-2.31.so              [.] __libc_enable_asynccancel
   0,07%  [kernel]                  [k] __x64_sys_read
   0,07%  libc-2.31.so              [.] __GI_____strtoull_l_internal
   0,06%  libc-2.31.so              [.] _int_malloc
   0,06%  [kernel]                  [k] number
   0,06%  libc-2.31.so              [.] __libc_disable_asynccancel
   0,06%  [kernel]                  [k] entry_SYSCALL_64_after_hwframe
   0,06%  [kernel]                  [k] pagecache_get_page
   0,05%  [kernel]                  [k] mutex_unlock
   0,05%  [kernel]                  [k] mutex_lock
   0,05%  [kernel]                  [k] __fdget_pos
   0,05%  libjava.so                [.] 0x00000000000160cc
   0,05%  libjvm.so                 [.] 0x00000000008d8da0
```

Life is better with symbols

The same program, but now it has a usable mapping from addresses to symbols, provided to us by the JIT compiler.

We see all the time is spent in I/O, in `java/io/BufferedReader;::readLine`

Let’s revisit this one

We’ll run this through **perf top**

```
CLASS Read_Text_Oneline2

PROPERTIES INHERITABLE
aL = ArrayList()

  METHOD Read_Text_Oneline2()
    RexxIO().File('./data/kjvdat.txt').forEachline(THIS.docid())

  METHOD main(args=String[]) STATIC
    r=Read_Text_Oneline2()
    LOOP FOR 10000
      LOOP i=0 TO r.aL.size()-1
        a = ChapterAndVerse r.aL.get(i)
        IF a.book='Rev' THEN
          IF a.chapter='22' THEN
            IF a.verse='21' THEN LEAVE
      END
    END
    SAY a

  CLASS Read_Text_Oneline2.docid DEPENDENT IMPLEMENTS LineHandler
  METHOD handle(in)

    PARSE in book '|' chapter '|' verse '|' textLine
    a = ChapterAndVerse(book,chapter,verse,textLine)
    PARENT.aL.add(a)
```

```
➜  bpfsprobes git:(master) perf stat java -XX:+PreserveFramePointer  Read_Text_Oneline2
Rev 22 21  The grace of our Lord Jesus Christ be with you all. Amen.~

 Performance counter stats for 'java -XX:+PreserveFramePointer Read_Text_Oneline2':

          27.941,67 msec task-clock                #    1,013 CPUs utilized
              6.797      context-switches          #    0,243 K/sec
              1.083      cpu-migrations            #    0,039 K/sec
             92.130      page-faults               #    0,003 M/sec
    110.927.691.194      cycles                    #    3,970 GHz
    240.753.224.323      instructions              #    2,17  insn per cycle
     46.301.682.611      branches                  # 1657,084 M/sec
         49.056.142      branch-misses             #    0,11% of all branches

       27,572957882 seconds time elapsed

       27,810831000 seconds user
        0,176500000 seconds sys
```

```
Samples: 50K of event 'cycles', 4000 Hz, Event count (approx.): 35775575473 lost: 0/0 drop: 0/0
Overhead  Shared Object             Symbol
  42,04%  [JIT] tid 102873          [.] LRead_Text_Oneline2;::main
  30,27%  [JIT] tid 102873          [.] Lnetrexx/lang/Rexx;::docompare
  18,86%  [JIT] tid 102873          [.] Lnetrexx/lang/Rexx;::OpAdd
   4,71%  [JIT] tid 102873          [.] Lnetrexx/lang/Rexx;::<init>
   0,25%  perf                      [.] __symbols__insert
   0,24%  perf                      [.] d_print_comp_inner
   0,15%  perf                      [.] rb_next
   0,11%  perf                      [.] rust_demangle_callback
   0,09%  [kernel]                  [k] module_get_kallsym
   0,08%  [kernel]                  [k] number
   0,08%  perf                      [.] d_print_comp
   0,07%  [kernel]                  [k] kallsyms_expand_symbol.constprop.0
   0,07%  [kernel]                  [k] vsnprintf
   0,07%  perf                      [.] d_count_templates_scopes
   0,07%  libc-2.31.so              [.] __GI_____strtoull_l_internal
   0,06%  libc-2.31.so              [.] _int_malloc
   0,06%  libjvm.so                 [.] 0x0000000000a9dc50
   0,06%  perf                      [.] rb_insert_color
   0,05%  [kernel]                  [k] string_nocheck
   0,05%  [kernel]                  [k] do_syscall_64
   0,05%  [kernel]                  [k] format_decode
   0,04%  [kernel]                  [k] memcpy_erms
   0,04%  perf                      [.] dso__find_symbol
   0,04%  libc-2.31.so              [.] _IO_getdelim
   0,04%  libc-2.31.so              [.] __memset_avx2_unaligned_erms
   0,03%  libc-2.31.so              [.] __strcmp_avx2
   0,03%  libc-2.31.so              [.] __libc_calloc
   0,03%  [kernel]                  [k] syscall_return_via_sysret
   0,02%  libc-2.31.so              [.] _IO_feof
   0,02%  perf                      [.] d_name
   0,02%  [unknown]                 [.] 0000000000000000
   0,02%  perf                      [.] __symbols__insert.constprop.0
   0,02%  libc-2.31.so              [.] __memmove_avx_unaligned_erms
   0,02%  [kernel]                  [k] __sched_text_start
   0,02%  perf                      [.] rb_erase
   0,02%  [kernel]                  [k] clear_page_erms
   0,02%  [kernel]                  [k] entry_SYSCALL_64
   0,02%  perf                      [.] perf_evsel__parse_sample
   0,02%  libc-2.31.so              [.] sysmalloc
   0,02%  [kernel]                  [k] psi_task_change
   0,02%  [kernel]                  [k] timerqueue_add
   0,02%  perf                      [.] d_make_comp
   0,02%  [kernel]                  [k] update_iter
   0,02%  [kernel]                  [k] update_blocked_averages
   0,02%  perf                      [.] sort__dso_cmp
   0,02%  perf                      [.] hists__findnew_entry
   0,02%  [kernel]                  [k] s_next
   0,02%  libc-2.31.so              [.] __strlen_avx2
   0,01%  [kernel]                  [k] prepare_exit_to_usermode
   0,01%  [kernel]                  [k] s_show
   0,01%  [kernel]                  [k] native_irq_return_iret
   0,01%  perf                      [.] d_demangle_callback
```

Hmm

We see that most time is spent in Rexx;::docompare and Rexx;::OpAdd

Why would that be?

Well, the compare is because of the comparison of every ArrayList element, which contains an instance of ChapterAndVerse, to the 3 strings.

The OpAdd is the loop counter. In this case …

We don’t need decimal loop counters

I changed that as can be seen on the right, using the DO … BINARY block delimiter.

That alone shaves off about 20 seconds.

```
CLASS Read_Text_Oneline2b

PROPERTIES INHERITABLE
aL = ArrayList()

  METHOD Read_Text_Oneline2b()
    RexxIO().File('./data/kjvdat.txt').forEachLine(THIS.docid())

  METHOD main(args=String[]) STATIC
    r=Read_Text_Oneline2b()
    DO BINARY
      LOOP FOR 10000
        LOOP i=0 TO r.aL.size()-1
          a = ChapterAndVerse r.aL.get(i)
          IF a.book=='Rev' THEN
            IF a.chapter=='22' THEN
              IF a.verse=='21' THEN LEAVE
        END
      END -- LOOP FOR
    END
    SAY a

  CLASS Read_Text_Oneline2b.docid DEPENDENT IMPLEMENTS LineHandler
  METHOD handle(in)

    PARSE in book '|' chapter '|' verse '|' textLine
    a = ChapterAndVerse(book,chapter,verse,textLine)
    PARENT.aL.add(a)
```

```
➜  bpfsprobes git:(master) perf stat java -XX:+PreserveFramePointer  Read_Text_Oneline2b
Rev 22 21  The grace of our Lord Jesus Christ be with you all. Amen.~

 Performance counter stats for 'java -XX:+PreserveFramePointer Read_Text_Oneline2b':

          8.362,75 msec task-clock                #    1,027 CPUs utilized
               680      context-switches          #    0,081 K/sec
                55      cpu-migrations            #    0,007 K/sec
            16.418      page-faults               #    0,002 M/sec
    19.206.783.758      cycles                    #    2,297 GHz
    15.628.203.011      instructions              #    0,81  insn per cycle
     3.228.597.088      branches                  #  386,069 M/sec
         9.766.935      branch-misses             #    0,30% of all branches

       8,145946642 seconds time elapsed

       8,325632000 seconds user
       0,044008000 seconds sys
```

№ 3

Profiling Rexx BIFs

```
/* REXX */
OPTIONS LEVELB

/* SUBSTR */
SAY "Look for SUBSTR OK"
/* do 1000000 */
/* These from the Rexx book. */
/* say '|'substr('abc',2)'|' */
IF SUBSTR('ABC',2) \= 'BC'            THEN SAY 'FAILED IN TEST          1 '
IF SUBSTR('ABC',2,4) \= 'BC  '        THEN SAY 'FAILED IN TEST          2 '
IF SUBSTR('ABC',2,6,'.') \= 'BC....'  THEN SAY 'FAILED IN TEST          3 '
/* These from Mark Hessling. */
IF SUBSTR("FOOBAR",2,3) \=   "OOB"    THEN SAY 'FAILED IN TEST          4 '
/* say '|'substr('foobar',3)'|' */
IF SUBSTR("FOOBAR",3) \=   "OBAR"     THEN SAY 'FAILED IN TEST          5 '
IF SUBSTR("FOOBAR",3,6) \=   "OBAR  " THEN SAY 'FAILED IN TEST          6 '
IF SUBSTR("FOOBAR",3,6,'*') \=   "OBAR**"  THEN SAY 'FAILED IN TEST          7 '
IF SUBSTR("FOOBAR",6,3) \=   "R  "    THEN SAY 'FAILED IN TEST          8 '
IF SUBSTR("FOOBAR",8,3) \=   "   "    THEN SAY 'FAILED IN TEST          9 '
IF SUBSTR('1234567890',5) \= '567890' THEN SAY 'FAILED IN TEST          10 '
/* say '|'substr('1234567890',5)'|' */
IF SUBSTR('1234567890',6,6,'.') \= '67890.' THEN SAY 'FAILED IN TEST          11 '
IF SUBSTR('ABC',2,4,'.') \= 'BC..'    THEN SAY 'FAILED IN TEST          12 '
IF SUBSTR('ABCDEFGH',1,2,'.') \= 'AB' THEN SAY 'FAILED IN TEST          13 '
IF SUBSTR('ABCDEFGH',2,3,'É') \= 'BCD'THEN SAY 'FAILED IN TEST          14 '
IF SUBSTR("René Vincent Jansen",1,4,".") \= 'René' THEN SAY 'FAILED IN TEST
IF SUBSTR("René Vincent Jansen",6,7,"") \= 'Vincent' THEN SAY 'FAILED IN TEST
IF SUBSTR("12345678",5,6,"É") \= '5678éé' THEN SAY 'FAILED IN TEST          17 '
/* say substr("12345678",10,6,"É") */
/* if substr("12345678",10,6,"éé") \= 'éééééé' then say 'need exceptions for this
/* end */
SAY "SUBSTR OK"

RETURN
```

SUBSTR BIF Testcase

Note the Unicode testcases

SUBSTR*

Fastest execution recorded

(For cRexx, excluding RXC time)

4.72 msec **ooRexx**

1.36 msec **BREXX**

0.77 msec **Regina**

0.48 msec **cREXX** - Rexx Version

0.43 msec **cREXX** - RXAS version

* Unicode testcases skipped except for CREXX

```
Samples: 123K of event 'cycles', 4000 Hz, Event count (approx.): 29857751311 lost: 0
Overhead  Shared Object                      Symbol
  12,40%  libRexx.so.4                      [.] MemoryObject::newObject
   6,26%  libRexx.so.4                      [.] MemorySegmentSet::sweepSingleSegment
   5,30%  libRexx.so.4                      [.] RexxExpressionFunction::evaluate
   4,69%  libRexx.so.4                      [.] RexxInstruction::evaluateArguments
   4,19%  libRexx.so.4                      [.] StringUtil::substr
   3,49%  libRexx.so.4                      [.] RexxInstructionIf::execute
   3,32%  libRexx.so.4                      [.] numberStringScan
   3,21%  libc-2.31.so                      [.] __memset_avx2_unaligned_erms
   2,94%  libRexx.so.4                      [.] RexxBinaryOperator::evaluate
   2,85%  libRexx.so.4                      [.] RexxInteger::unsignedNumberValue
   2,58%  libRexx.so.4                      [.] RexxActivation::run
   2,57%  libRexx.so.4                      [.] ProtectedBase::ProtectedBase
   2,56%  libRexx.so.4                      [.] RexxString::comp
   2,50%  libRexx.so.4                      [.] builtin_function_SUBSTR
   2,18%  libc-2.31.so                      [.] toupper
   2,13%  libRexx.so.4                      [.] RexxString::stringComp
   1,85%  libRexx.so.4                      [.] ExpressionStack::requiredIntegerArg
   1,68%  libRexx.so.4                      [.] RexxObject::callOperatorMethod
   1,52%  libRexx.so.4                      [.] ExpressionStack::expandArgs
   1,43%  libc-2.31.so                      [.] __memmove_avx_unaligned_erms
   1,38%  libRexx.so.4                      [.] RexxString::evaluate
   1,33%  libc-2.31.so                      [.] __memcmp_avx2_movbe
   1,32%  libRexx.so.4                      [.] RexxString::numberString
   1,31%  libRexx.so.4                      [.] RexxInternalObject::requestString
   1,30%  libRexx.so.4                      [.] ExpressionStack::optionalIntegerArg
   1,28%  libRexx.so.4                      [.] positionArgument
   1,17%  libRexx.so.4                      [.] NumberString::parseNumber
   1,05%  libRexx.so.4                      [.] ExpressionStack::requiredStringArg
   0,97%  libRexx.so.4                      [.] RexxInteger::evaluate
   0,97%  libRexx.so.4                      [.] NumberString::newInstance
   0,85%  libRexx.so.4                      [.] lengthArgument
   0,80%  libRexx.so.4                      [.] RexxInternalObject::requiredString
   0,80%  libRexx.so.4                      [.] RexxString::rawString
   0,75%  libRexx.so.4                      [.] RexxInternalObject::isBaseClass
   0,70%  libRexx.so.4                      [.] NumberString::comp
```

CPU Profile of ooRexx substr

1) We need to run this in a loop to see significant CPU usage

2) The memory overhead might be of that loop

3) Still, we clearly see the relative CPU profile of the called functions

SUBSTR in RXAS

Well, 3/4 of it

Based on algorithm in ANSI standard

```
/*
 * REXX SUBSTR BIF IN RXAS. USES LENGTH.RXAS
 */
          .GLOBALS=0
LENGTH()  .EXPOSE=LENGTH.LENGTH
SUBSTR()  .LOCALS=18 .EXPOSE=SUBSTR.SUBSTR
    /* STRING    = ARG1    */
    /* N (START) = ARG2    */
    LOAD  R7," " /* THE DEFAULT PAD CHARACTER */
    LOAD  R6,0   /* THE LENGTH FIELD INITIALIZED TO 0 */
    BRTPANDT HAVETHREE,A3,1
    ISUB  A0,A0,1
HAVETHREE:
    BRTPANDT HAVEFOUR,A4,1
    ISUB  A0,A0,1
HAVEFOUR:
    /* NOW WE HAVE THE CORRECT NUMBER */
    ILT   R1,A0,2 /* LESS THAN 2 ARGUMENTS */
    BRT   WRONGARGS,R1
    IGT   R1,A0,4 /* MORE THAN 4 ARGUMENTS */
    BRT   WRONGARGS,R1
    IEQ   R1,A0,2
    BRT   DOIT,R1 /* WE HAVE TWO ARGUMENTS */
    COPY  R6,A3 /* SAVE LENGTH FIELD IN R6*/
    IEQ   R1,A0,4
    BRF   DOIT,R1 /* WE DO NOT HAVE 4 ARGUMENTS, NO PAD */
    COPY  R7,A4  /* WE HAVE A PAD, REPLACE THE DEFAULT ONE */
    BR    DOIT
WRONGARGS:
    /* WE ARE HERE IF THERE ARE NOT ENOUGH, OR TOO MANY ARGUMENTS */
    SAY "SUBSTR NEEDS AT LEAST TWO AND AT MOST 4 ARGUMENTS"
    EXIT
DOIT:
    /* WE WANT TO KNOW THE LENGTH OF THE STRING ARGUMENT */
    LOAD  R3,1 /* THERE IS ONE ARGUMENT FOR THE CALL TO LENGTH() */
    COPY  R4,A1 /* AND IT IS THE STRING IN A1 */
    CALL  R10,LENGTH(),R3 /* WHAT IS THE LENGTH OF THE STRING ARG */
    ITOS R10
    COPY  R12,R10 /* THE PRELIMINARY LOOP COUNTER IS THE STRLENGTH */
    DEC   A2
    IGT   R1,R6,0
    BRF   SKIP,R1
    COPY  R12,R6 /* THE FINAL LOOP COUNTER IS THE LENGTH ARGUMENT */
    IADD  R12,R12,A2 /* ADAPT FOR START POSITION */
SKIP:
    IGT   R17,A2,R10 /* IF THE START POS IS GREATER THAN STRING LENGTH */
    BRT   PAD,R17
```

SUBSTR in level B cREXX

.. for an impression, all the code is in:

https://github.com/adesutherland/CREXX

The clarity of this, coupled with the almost not measurable performance difference, made us decide to implement most BIF's in Rexx.

(Which Peter subsequently did).

```
/* Classic REXX Runtime Library */

OPTIONS LEVELB /* Written in REXX Level B */

/* declarations */
/* Raise() Internal Function to Raise a runtime error */
RAISE: PROCEDURE = .INT
  ARG TYPE = .STRING, CODE = .STRING, PARM1 = .STRING

/* Length() Procedure */
LENGTH: PROCEDURE = .INT
  ARG STRING1 = .STRING

/* Substr() Procedure */
SUBSTR: PROCEDURE = .STRING
  ARG STRING1 = .STRING, START = .INT, LEN = LENGTH(STRING1) + 1 - START, PAD = ' '

  PADCHAR = 0 /* Is an integer */
  OUTPUT = ''
  INPUTLENGTH = 0;
  PADLENGTH = 0;

  /* Check Start */
  IF START < 1 THEN CALL RAISE "SYNTAX", "40.13", START /* Invalid start */
  START = START - 1 /* 1 base to zero base */

  /* Check Length */
  IF LEN < 1 THEN CALL RAISE "SYNTAX", "40.13", LEN /* Invalid start */

  /* Check length of pad */
  ASSEMBLER STRLEN PADLENGTH,PAD;
  IF PADLENGTH > 1 THEN CALL RAISE "SYNTAX", "40.23", PAD /* Invalid pad length */

  /* Get the Length of the input string */
  ASSEMBLER STRLEN INPUTLENGTH,STRING1
  INPUTLENGTH = INPUTLENGTH - START;

  IF INPUTLENGTH > 0 THEN DO
    /* Yes there are characters needed from string1 */
    IF LEN <= INPUTLENGTH THEN DO
      /* Just copy from string1 - no padding needed */
      DO I = START TO START + LEN - 1
        ASSEMBLER CONCCHAR OUTPUT,STRING1,I
      END
    END
    ELSE DO
      /* Copy all of string1 and then pad */
      DO I = START TO START + INPUTLENGTH - 1
        ASSEMBLER CONCCHAR OUTPUT,STRING1,I
      END

      /* Then add pads */
```

№ 4

CPS: The Clauses Per Second Benchmark

ARM is on the move

The latest from MFC's Speleotrove

Date	RexxCPS	Hardware	Software environment				
2021.10.28	23,774,392	M1 Mac ARM 64	Darwin ooRexx_5.0.0	6.05	14	Sep	2021
2015.03.06	19,413,819	IBM z13	CMS REXXC370	4.02	23	Dec	1999
2013.07.04	17,778,252	IBM zEC12 2827-789	CMS REXXC370	4.02	23	Dec	1999
2021.10.28	15,928,590	M1 Mac ARM 64	Unix Regina 3.9.3	5.00	5	Oct	2019
2012.01.01	14,766,746	Intel i5 2.5 GHz	Win7 DosCrx1.0	5.00	22	Apr	1999
2021.08.30	14,418,411	iMac Apple Silicon M1	Darwin	6.05	12	Aug	2021
2011.06.00	14,126,688	IBM z196 2817-742	CMS REXXC370	4.02	23	Dec	1999
2020.06.14	12,500,000	Lenovo T540-15ICB	Win10-64 ooRexx 4.2.0	6.04	22	Feb	2014
2020.01.27	11,494,253	Lenovo T540-15ICB	Win10-64 Regina 3.9.3	5.00	5	Oct	2019
2011.06.08	10,135,135	Intel i7 4.7 GHz	Win7 ooRexx 4.1.0	6.03	5	Dec	2010
2014.05.05	8,287,671	Pentium G3220 3 GHz	Win7-64 Regina 3.7	5.00	?		
2014.01.08	7,665,816	Intel Xeon 3.5 GHz	Win7 ooRexx 4.1.2	6.03	28	Aug	2012
2012.05.25	6,675,567	Intel i5 2.5 GHz	Win7 ooRexx 4.1.0	6.03	5	Dec	2010
2012.03.26	6,192,687	Xeon 3.1GHz 4-way	Linux jREXX	0.0.3	26	Mar	2012
2001.08.09	5,567,929	AMD Athlon 1.4 GHz	DosCrx1.0 16 bit	5.0	2	Dec	1999

```
➜  test git:(master) X ooRexx rexxcp
----- REXXCPS 2.1 -- Measuring REXX
 REXX version is: REXX-ooRexx_5.0.0(
       System is: DARWIN
       Averaging: 100 measures of 10

Total (full DO): 0.00446599 secs (av
Time for one iteration (1000 clauses

     Performance: 22391452 REXX clau

➜  test git:(master) X
```

```
➜  test git:(master) X regina rexxcps.
----- REXXCPS 2.1 -- Measuring REXX cl
 REXX version is: REXX-Regina_3.9.3(MT
       System is: UNIX
       Averaging: 100 measures of 100

Total (full DO): 0.00670827 secs (aver
Time for one iteration (1000 clauses)

     Performance: 14906973 REXX clause

➜  test git:(master) X
```

```
➜  test git:(master) ooRexx rexxcps.rex
----- REXXCPS 2.1 -- Measuring REXX clauses
 REXX version is: REXX-ooRexx_5.0.0(MT)_64-
       System is: LINUX
       Averaging: 100 measures of 100 itera

Total (full DO): 0.00738375 secs (average o
Time for one iteration (1000 clauses) was:

     Performance: 13543254 REXX clauses per

➜  test git:(master)
```

```
➜  test git:(master) rexx rexxcps.rex
----- REXXCPS 2.1 -- Measuring REXX claus
 REXX version is: REXX-Regina_3.9.4 5.00
       System is: UNIX
       Averaging: 100 measures of 100 ite

Total (full DO): 0.01171962 secs (average
Time for one iteration (1000 clauses) was

     Performance: 8532700 REXX clauses pe

➜  test git:(master)
```

Platforms

Different platforms, different scores

Need a performance regression section on the RexxLA Jenkins build machine.

The End. For the moment.