Using VTune for athena profiling

G. Gaycken

CERN, 2024 November 22

About

- **VTune**: profiling tool developed by Intel for x86_64 (Linux, Win, Mac)
- on "modern" Intel CPUs can use hardware counters, on other platforms time only.
- sampling used to associate counts or time to code(-sections), and/or alternate counters if more counters are active than can be monitored simultaneously.
- on Linux the majority(all?) of profile data can also be collected with Linux "perf", but
- VTune can handle much better large profiles (e.g. on a 32 GB system it is basically impossible to profile athena with Linux perf, but it works reasonably well with VTune)
- For modern AMD CPUs there is uProf

Prerequisites (1)

- VTune is setup (→ Installation guide). At CERN:

```
source /cvmfs/projects.cern.ch/intelsw/oneAPI/linux/all-setup.sh
```

- Optional and not used here but recommended for athene development. Atlas git extension setup:

```
lsetup git
```

- athena is setup

```
asetup Athena,main,latest
```

Prerequisites (2)

- Optional but recommended: the athena package **PerfMonVTune** is checked-out and compiled (package in the athena repository but not part of e.g. the Athena project). (more instructions: → ATLAS software docs)

```
git clone --no-checkout https://gitlab.cern.ch/atlas/athena/
cd athena
git config core.sparsecheckout true # to avoid checking out all 65k files
git branch  acts_vtune nightly/main/2024-11-11T2101
git ls-tree -r acts_vtune | grep "VTune" # to uncover the package name
echo Projects/WorkDir/ >> .git/info/sparse-checkout
echo Control/PerformanceMonitoring/PerfMonVTune/ >> .git/info/sparse-checkout
echo Tracking/Acts/ActsTrackReconstruction/ >> .git/info/sparse-checkout
git checkout -f
mkdir -p ../build && cd ../build/ && cmake ../athena/Projects/WorkDir/ && make
source $Athena_PLATFORM/setup.sh
```

Prerequisites (2)

- Optional but recommended: postInclude to only collect during algorithm execution. vtune.py:

```
def vtune12(flags, cfg) :
    from AthenaConfiguration.ComponentFactory import CompFactory
    cfg.addService ( CompFactory.VTuneProfilerService(
          "VTuneProfilerService",
          ResumeEvent = 2,
          PauseEvent = 10
        ),
        create=True)
```

Create athena config

Create a pickle file to avoid profiling the python based configuration step:

... --config-only=rec_vtune_test.pkl ...

```
  export ATHENA_CORE_NUMBER=2
  Reco_tf.py \
 --maxEvents '12' \
 --perfmon 'fastmonmt' \
 --multithreaded 'True' \
 --autoConfiguration 'everything' \
 --conditionsTag 'all:OFLCOND-MC21-SDR-RUN4-02' \
 --geometryVersion 'all:ATLAS-P2-RUN4-03-00-00' \
 --postInclude 'all:PyJobTransforms.UseFrontier,vtune.vtune12' \
 --preInclude 'InDetConfig.ConfigurationHelpers.OnlyTrackingPreInclude,ActsConfig.ActsCIFlags.actsAloneWorkflowFlags' \
 --steering 'doRAWtoALL' \
 --preExec 'all:flags.Exec.FPE=-1;ConfigFlags.Tracking.doITkFastTracking=False' \
 --postExec 'all:cfg.getService("AlgResourcePool").CountAlgorithmInstanceMisses␣=␣True;' \
 --inputRDOFile '/cvmfs/atlas-nightlies.cern.ch/repo/data/data-art/PhaseIIUpgrade/RDO/ATLAS-P2-RUN4-03-00-00/'\
'mc21_14TeV.601229.PhPy8EG_A14_ttbar_hdamp258p75_SingleLep.recon.RDO.e8481_s4149_r14700/*' \
 --outputAODFile 'myAOD.pool.root' \
 --athenaopts="--config-only=rec_vtune_test.pkl"
```

Running VTune on Intel

hotspot collection, start paused because VTuneProfilerService will resume before and pause after algorithm execution. Without VTuneProfilerService huge profile about the initialization and finalization.

```
   vtune \
  -data-limit=3000 \
  -collect hotspots \
  -start-paused \
  -mrte-mode=native  -knob sampling-mode=hw \
  -knob sampling-interval=0.5 -knob enable-stack-collection=true \
  -strategy=':trace:trace,ld-linux.so.2:notrace:notrace,ld-2.12.so'\
':notrace:notrace,ld-linux.so:notrace:notrace,ld-linux-x86-64.so.2:'\
'notrace:notrace' \
  -- $(which python) $(which athena.py) \
  --preloadlib=$ATLASMKLLIBDIR_PRELOAD/libintlc.so.5\
:$ATLASMKLLIBDIR_PRELOAD/libimf.so \
 rec_vtune_test.pkl  > log.vtune 2>&1
```

Running VTune on AMD

On AMD only timing information collected

```
   vtune \
  -data-limit=3000 \
  -collect hotspots \
  -start-paused \
  -strategy=':trace:trace,ld-linux.so.2:notrace:notrace,ld-2.12.so'\
':notrace:notrace,ld-linux.so:notrace:notrace,ld-linux-x86-64.so.2:'\
'notrace:notrace' \
  --  $(which python) $(which athena.py) \
  --preloadlib=$ATLASMKLLIBDIR_PRELOAD/libintlc.so.5\
:$ATLASMKLLIBDIR_PRELOAD/libimf.so \
 rec_vtune_test.pkl  > log.vtune 2>&1
```

Analysing VTune results

- VTune GUI:

  vtune-gui r*/r*.vtune

- web-interface:
  Start the back-end using a port e.g. 8081:

  vtune-backend --web-port 8081 --reset-passphrase --data-directory ./

  If outside of CERN start ssh tunnel with forward to node ??? on which VTune was running using e.g. port 2029:

  ssh -L 2029:lxplus???:22 -N lxtunnel.cern.ch

  Then tunnel to special node (e.g. via CERN tunnel) :

  ssh -L 8081:localhost:8081 -N -p 2029 localhost

  Finally use a browser to connect to the web server as suggested by VTune.

Profiling options

There are multiple data collection options in VTune (e.g. -collect threading) (→ full list)

- hotspots: execution time
- threading: e.g. waiting on locks
- memory-consumption (huge profiles, should limit number of events)
- uarch-exploration: port utilization (Intel only); incompatible with “-knob sampling-mode=hw“; requires /proc/sys/kernel/perf_event_paranoid==0

Kernel tweaks for advanced profiling

- **Sampling driver** for instruction level precision
- For uarch-exploration, kernel profiling:
  permissive: (requires root)

  echo 0 > /proc/sys/kernel/kptr_restrict
  echo 0 > /proc/sys/kernel/perf_event_paranoid

  typical more restrictive settings:

  echo 1 > /proc/sys/kernel/kptr_restrict
  echo 2 > /proc/sys/kernel/perf_event_paranoid