CO339 – Microbenchmarking Tutorial

Holger Pirk

Welcome to the CO 339 tutorial on microbenchmarking. We are going to practice the development of useful microbenchmarks from complex applications.

1 Before we start

Claim a machine by entering your name behind a machine name `https://tinyurl.com/y6rfy5fv`

Here is a nice line to get a temporary environment `cd $(mktemp -d)`

1.1 Create some sample data

(This will take some time so we run it in the background)

```
curl https://www.doc.ic.ac.uk/~hlgr/randomnumbers.cpp | clang++ -O3 -x c++ - && ./a.out > /tmp/points.csv &
```

2 Setup

Now, we build and install MonetDB (an open-source, in-memory database management system)

```
cd $(mktemp -d)
curl -L https://www.monetdb.org/downloads/sources/Oct2020-SP2/MonetDB-11.39.11.tar.bz2 | tar xj
cd ./MonetDB-11.39.11/
curl -L http://www.doc.ic.ac.uk/~hlgr/no_imprints.patch | patch -p1
mkdir Release
cd Release
cmake -DCMAKE_BUILD_TYPE=RelWithDebInfo  -DRINTEGRATION=NO -DGEOM=NO -DFITS=NO -DNETCDF=NO -DODBC=NO -DPY3INTEGRATION=NO
↪  -DWITH_CURL=NO -DBUILD_TESTING=NO -DWITH_XML2=NO -DINT128=NO -DCMAKE_INSTALL_PREFIX=$PWD/../.. ..
cmake --build . -j --target install
cd ../..
./bin/monetdbd create dbfarm
./bin/monetdbd start dbfarm
./bin/monetdb create test
./bin/monetdb release test
echo user=monetdb > ~/.monetdb
echo password=monetdb >> ~/.monetdb
./bin/mclient test
```

Now, we load the data we created

```
create table points (x int not null, y int not null);
COPY 134217728 RECORDS INTO points from '/tmp/points.csv' USING DELIMITERS ',','\n' LOCKED;
\q
```

Now, we set the database to read-only (this simplifies the analysis)

```
./bin/monetdb stop test
./bin/monetdb set readonly=yes test
```

3 First task

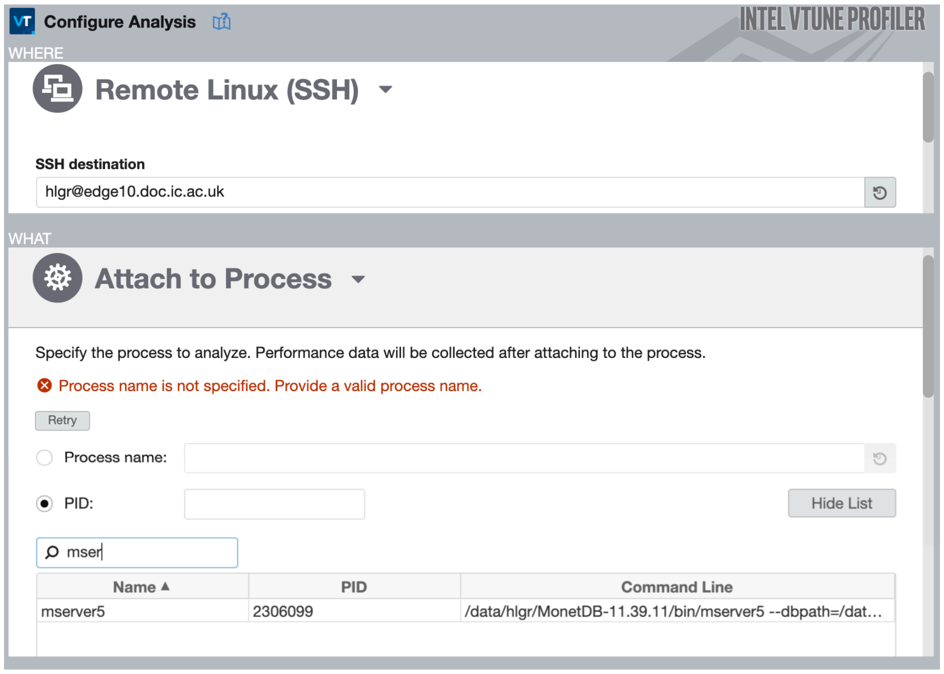
Now, we get started

3.1 Let's look at the sensitivity to the query parameters

```
for s in {1..10}; do (for i in {1..3}; do ./bin/mclient -f csv test -s "set optimizer='sequential_pipe'; trace select count(*)
↪  from points where x<=$[50*s];" | grep algebra.thetaselect | cut -d, -f 1; done) | awk '{s+=$0} END {print s/3}'; done
```

3.2 Let's look at a query through VTune

1. Find the process and attach vtune To make sure you profile the right process in vtune, select "Attach to Process" in the "WHAT" section of the "Configure Analysis" dialog, select "PID" and press the "SELECT" button. In the displayed list, you search for "mserver5" and double-click the process. That should fill the "PID" field.



SSH destination: hlgr@edge10.doc.ic.ac.uk

Specify the process to analyze. Performance data will be collected after attaching to the process.

Process name is not specified. Provide a valid process name.

Process name:

PID:

mser

Name ▲	PID	Command Line
mserver5	2306099	/data/hlgr/MonetDB-11.39.11/bin/mserver5 --dbpath=/dat…

2. Run a query

```
./bin/mclient test -s "set optimizer='sequential_pipe'; select count(*) from points where x<=250;"
```

3. Look at the profile to nail down the hot path

   - Drill down to the hot lines of code
   - you might want to download the source code onto your machine:

```
curl -L https://www.monetdb.org/downloads/sources/Oct2020-SP2/MonetDB-11.39.11.tar.bz2 | tar xj
```

   - At this point, take a break and let us meet in the main room

3.3 Cheat

- For reference, here is a line you might not understand right now – I will explain

```
cd gdk && cp gdk_select.c gdk_select_bak.c && cc -DLIBGDK -D_GNU_SOURCE -D_XOPEN_SOURCE -Dbat_EXPORTS -I../common/stream
↪  -I../common/utils -I../gdk -I../Release -I../common/options  -DNDEBUG=1 -DNDEBUG   -std=gnu99 -E gdk_select_bak.c | egrep
↪  -v '^# [0-9]+ ' | indent > gdk_select.c
```

or download the code from `https://www.doc.ic.ac.uk/~hlgr/gdk_select.c`

3.4 Let's set up a microbenchmark

I have created a prototype for a microbenchmark for you. Here is how you get started:

```
git clone https://gitlab.doc.ic.ac.uk/pe/MicroMonet.git # use your doc-login
cd MicroMonet
mkdir build
cd build
CC=clang CXX=clang++ cmake -DCMAKE_BUILD_TYPE=Release ..
make -j$(nproc)
./MicroMonet
nano ../MicroMonet.cpp # nano or your editor of choice
```

3.5 Objectives

- Develop a microbenchmark that models the behavior of the relevant monetdb operator(s) in `MicroMonet.cpp` (see the comments for guidance)
- Describe the access pattern of your operator in the access pattern algebra
- What is a lower runtime bound for your implementation based on the memory bandwidth
  - You need to calculate the memory bandwith (check out Intel's ARK for information about your CPU)
- Compare the lower bound with your implementation's performance as well as that of MonetDB
- Let's Discuss
- Remember to put in the appropriate "do not optimize" statements to avoid dead-code-elimination (if you do not use a variable, the compiler may remove it)

3.6 To find out your CPU model

```
cat /proc/cpuinfo  | grep "model name" | uniq
```

4 Second task (this one is optional)

Let's do the same thing for a more complex query: let's keep `x<=250` and introduce a varying parameter for `y`

```
./bin/mclient -f csv test        -s "set optimizer='sequential_pipe'; trace select count(*) from points where x<=250 and
↪  y<=250;"
```

4.1 Same spiel, what changed?

- Do we need a new
  - Microbenchmark?
  - Access Pattern?
  - Lower bound
- Let's discuss some more