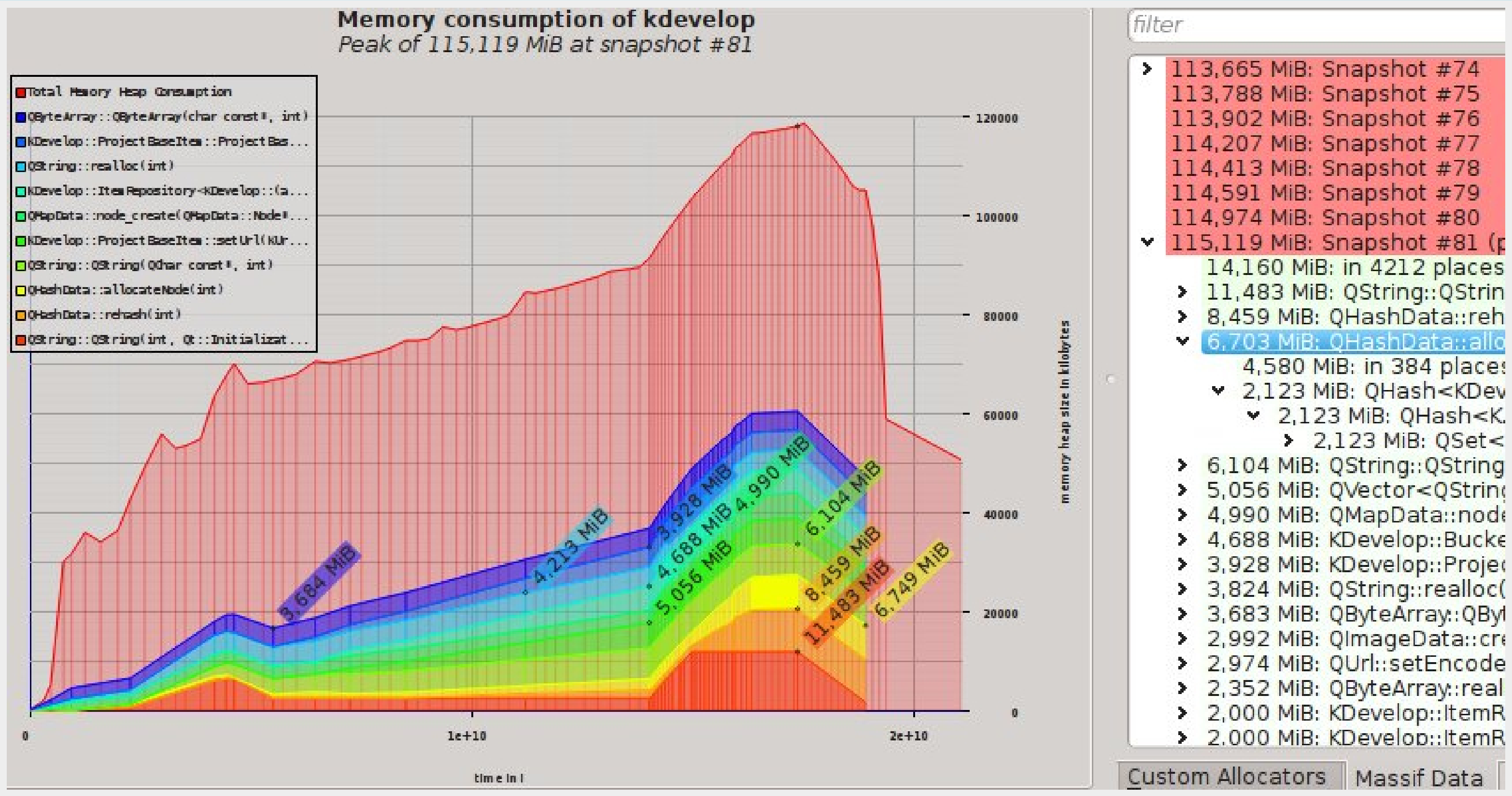
HEAPTRACK

A HEAP MEMORY PROFILER FOR LINUX

Thomas McGuire / Milian Wolff

www.kdab.com

MEMORY PROFILING


Memory consumption of kdevelop
Peak of 115,119 MiB at snapshot #81
🟥 Total Memory Heap Consumption
🟦 QByteArray::QByteArray(char const*, int)
🟦 KDevelop::ProjectBaseItem::ProjectBas...
🟦 QString::realloc(int)
🟩 KDevelop::ItemRepository<KDevelop::(a...
🟩 QMapData::node_create(QMapData::Node*...
🟩 KDevelop::ProjectBaseItem::setUrl(KUr...
🟩 QString::QString(QChar const*, int)
🟨 QHashData::allocateNode(int)
🟧 QHashData::rehash(int)
🟥 QString::QString(int, Qt::Initializat...
3,684 MiB
4,213 MiB
3,928 MiB
4,688 MiB
5,056 MiB
4,990 MiB
6,104 MiB
8,459 MiB
11,483 MiB
6,749 MiB
120000
100000
80000
60000
40000
20000
0
memory heap size in kilobytes
0
1e+10
2e+10
time in i
filter
Custom Allocators
Massif Data

```
›  113,665 MiB: Snapshot #74
   113,788 MiB: Snapshot #75
   113,902 MiB: Snapshot #76
   114,207 MiB: Snapshot #77
   114,413 MiB: Snapshot #78
   114,591 MiB: Snapshot #79
   114,974 MiB: Snapshot #80
⌄  115,119 MiB: Snapshot #81 (p
       14,160 MiB: in 4212 places
   ›   11,483 MiB: QString::QStrin
   ›   8,459 MiB: QHashData::reh
   ⌄   6,703 MiB: QHashData::allo
           4,580 MiB: in 384 places
       ⌄   2,123 MiB: QHash<KDev
           ⌄   2,123 MiB: QHash<K
               ›   2,123 MiB: QSet<
   ›   6,104 MiB: QString::QString
   ›   5,056 MiB: QVector<QString
   ›   4,990 MiB: QMapData::node
   ›   4,688 MiB: KDevelop::Bucke
   ›   3,928 MiB: KDevelop::Projec
   ›   3,824 MiB: QString::realloc(
   ›   3,683 MiB: QByteArray::QByt
   ›   2,992 MiB: QImageData::cre
   ›   2,974 MiB: QUrl::setEncode
   ›   2,352 MiB: QByteArray::real
   ›   2,000 MiB: KDevelop::ItemR
   ›   2,000 MiB: KDevelop::ItemR
```

THE GOAL

- a faster massif
- more information than massif
- research project

HOW?

- **`LD_PRELOAD`** overloading of **`malloc`** & friends
- libunwind for fast stack traces
- out-of-process analysis of DWARF information
- on-demand interpretation of the data

INSTALLATION

```
git clone git://anongit.kde.org/scratch/mwolff/heaptrack
cd heaptrack
mkdir build
cd build
cmake ..
make install
```

RUNNING

```
$ heaptrack someapp
starting application, this might take some time...
output will be written to /home/milian/heaptrack.someapp.1234.gz
...
Heaptrack finished! Now run the following to investigate the data:

  heaptrack_print /home/milian/heaptrack.someapp.1234.gz | less
```

ANALYSIS

```
heaptrack_print /home/milian/heaptrack.test_cpp.1234.gz | less
...
MOST CALLS TO ALLOCATION FUNCTIONS
11 calls to allocation functions with 44 bytes peak consumption from
0x400e27 /tmp/test_cpp asdf() /tmp/test.cpp:18
  10 calls with 40 bytes peak consumption from:
    0x400e59 /tmp/test_cpp bar() /tmp/test.cpp:25
    0x400e69 /tmp/test_cpp laaa() /tmp/test.cpp:30
    0x4010a2 /tmp/test_cpp main /tmp/test.cpp:66
  1 calls with 4 bytes peak consumption from:
    0x400e59 /tmp/test_cpp bar() /tmp/test.cpp:25
    0x400e69 /tmp/test_cpp laaa() /tmp/test.cpp:30
    0x4010ce /tmp/test_cpp main /tmp/test.cpp:69
...
```

QT ISSUE FOUND

```
QTextStream out;
for (int i = 0; i < 100; ++i) {
  out << 'i' << i << '\n';
}
```

ANALYSIS

```
heaptrack_print /home/milian/heaptrack.kate.1234.gz | less
...
PEAK MEMORY CONSUMERS
1048576 bytes peak memory consumed over 4 calls from
0x7f078ff83cf8 /usr/lib/libQtCore.so.4 QString::realloc(int)
0x7f078ff87463 /usr/lib/libQtCore.so.4 QString::append(QString const&)
0x7f078ffd97e5 /usr/lib/libQtCore.so.4 ??
0x7f078ffd9db7 /usr/lib/libQtCore.so.4 ??
0x7f078ffdabcc /usr/lib/libQtCore.so.4 QTextStream::readAll()
0x7f078027a6fb /usr/lib/libkatepartinterfaces.so.4
               Kate::Script::readFile(QString const&, QString&)
               /home/milian/kate/part/script/katescripthelpers.cpp:52
0x7f078027ab41 /usr/lib/libkatepartinterfaces.so.4
               Kate::Script::require(QScriptContext*, QScriptEngine*)
               /home/milian/kate/part/script/katescripthelpers.cpp:121
...
```

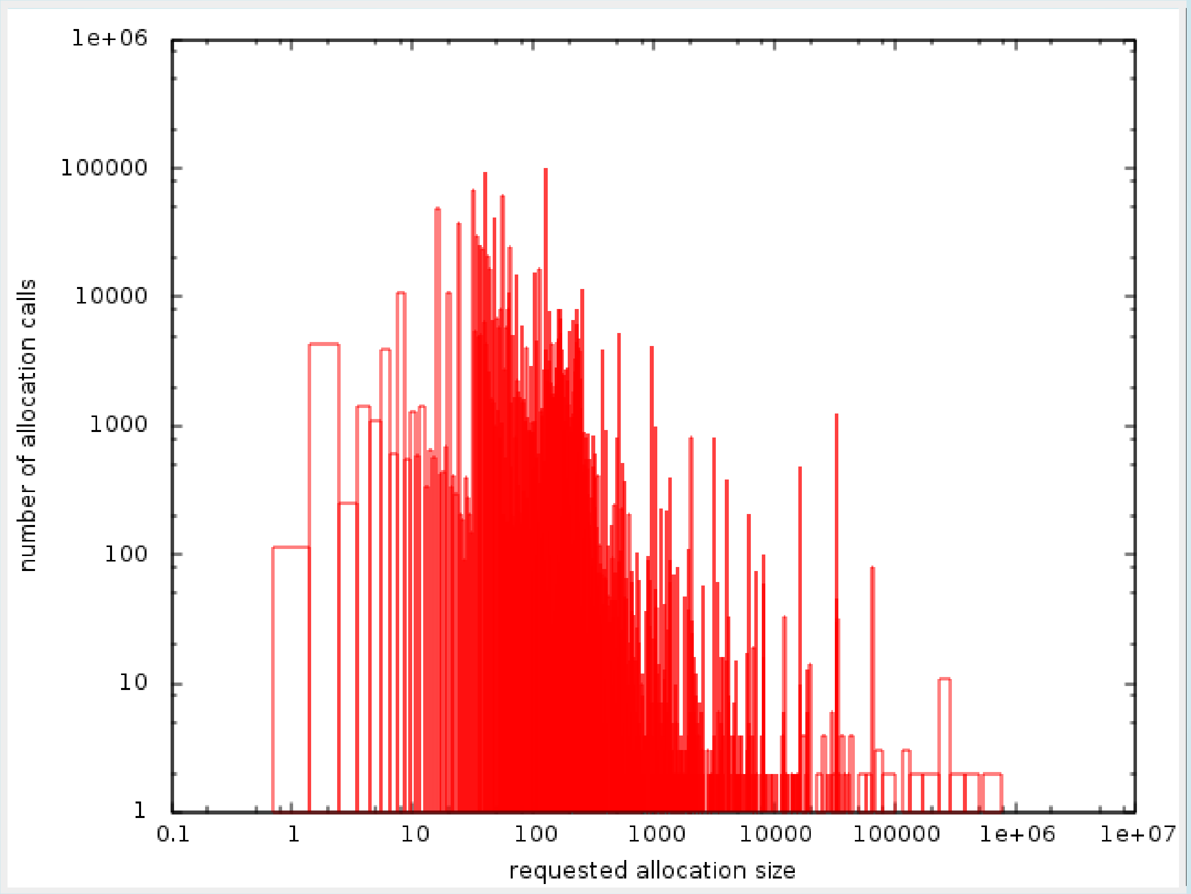
ANALYSIS

```
heaptrack_print -l=1 /home/milian/heaptrack.kate.1234.gz | less
...
MEMORY LEAKS
8192 bytes leaked over 1 calls from
0x7f0790837a8d /usr/lib/libkdecore.so.5
               KZoneAllocator::allocate(unsigned long)
0x7f0791cb7984 /usr/lib/libkdeui.so.5
               KCompTreeNode::insert(QChar const&, bool)
0x7f0791cb7acf /usr/lib/libkdeui.so.5
               KCompletion::addItem(QString const&, unsigned int)
0x7f0791cb7c9d /usr/lib/libkdeui.so.5
               KCompletion::insertItems(QStringList const&)
0x7f07803af80e /usr/lib/libkatepartinterfaces.so.4
               KateCmd::registerCommand(KTextEditor::Command*)
               /home/milian/kate/part/utils/katecmd.cpp:56
...
```

ANALYSIS

```
heaptrack_print -o=1 /home/milian/heaptrack.kate.1234.gz | less
...
MOST BYTES ALLOCATED OVER TIME (ignoring deallocations)
14516224 bytes over 443 calls allocated
0x7f078ff3bda6 /usr/lib/libQtCore.so.4
               QByteArray::realloc(int)
0x7f078ff3beff /usr/lib/libQtCore.so.4
               QByteArray::resize(int)
0x7f078ffca35a /usr/lib/libQtCore.so.4
               QIODevice::readLine(long long)
0x7f079079767e /usr/lib/libkdecore.so.5
               KMimeTypeRepository::parseMagicFile(...)
0x7f07907987ec /usr/lib/libkdecore.so.5
               KMimeTypeRepository::parseMagic()
...
```

ANALYSIS


1e+06
100000
10000
1000
100
10
1
number of allocation calls
0.1
1
10
100
1000
10000
100000
1e+06
1e+07
requested allocation size

THE GOOD

- useable
- often faster than massif
- more data to interpret
- foundation for more tools

THE BAD

- proper GUI missing
- custom allocators

THE UGLY

- Linux platform specific
- stack memory

THE END

TIME TO CAST YOUR VOTE!

Thomas McGuire / Milian Wolff

www.kdab.com

```
git clone git://anongit.kde.org/scratch/mwolff/heaptrack
```