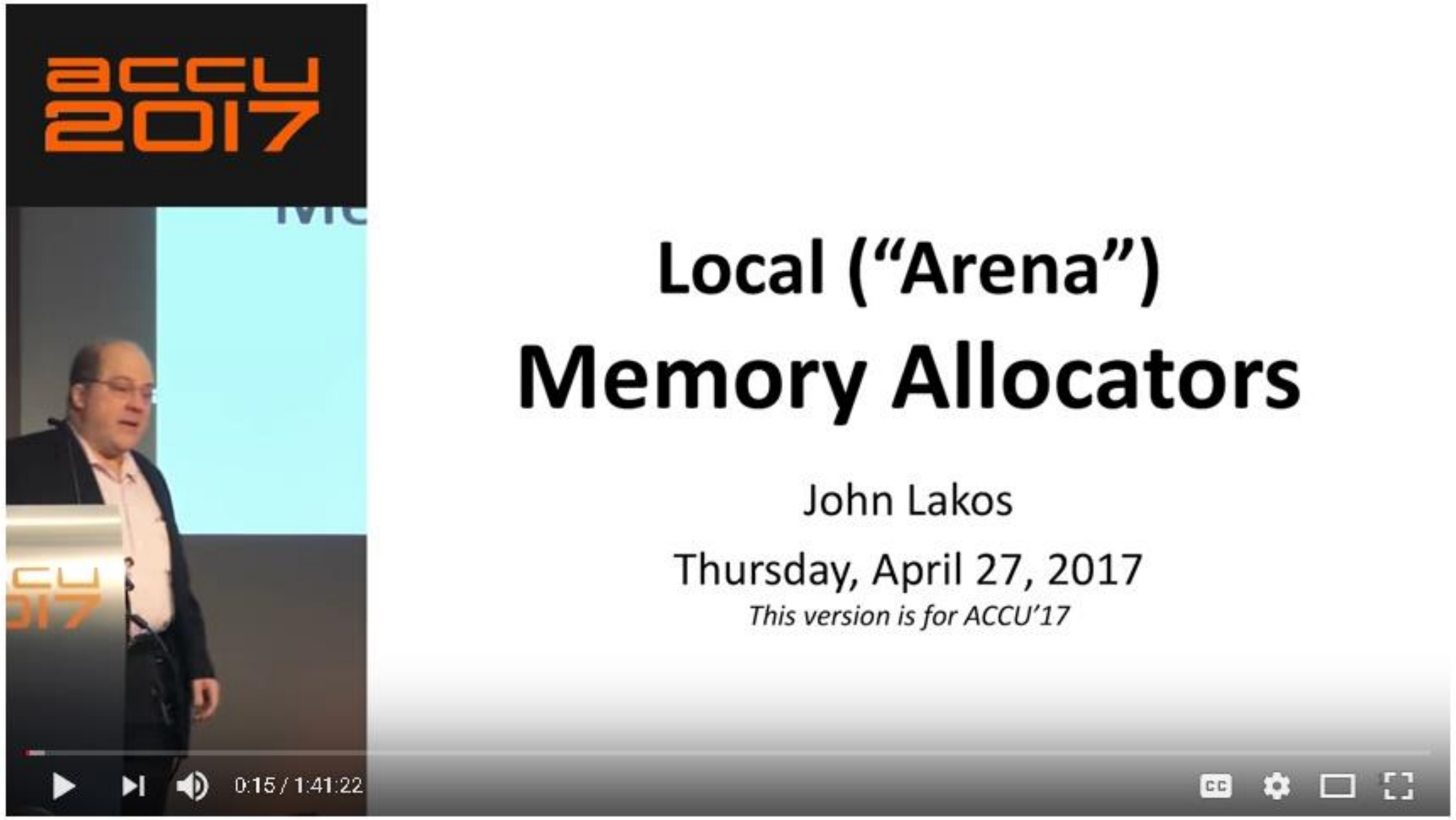
Reducing memory allocations

Arnaud Desitter
ACCU conference - 13 April 2018

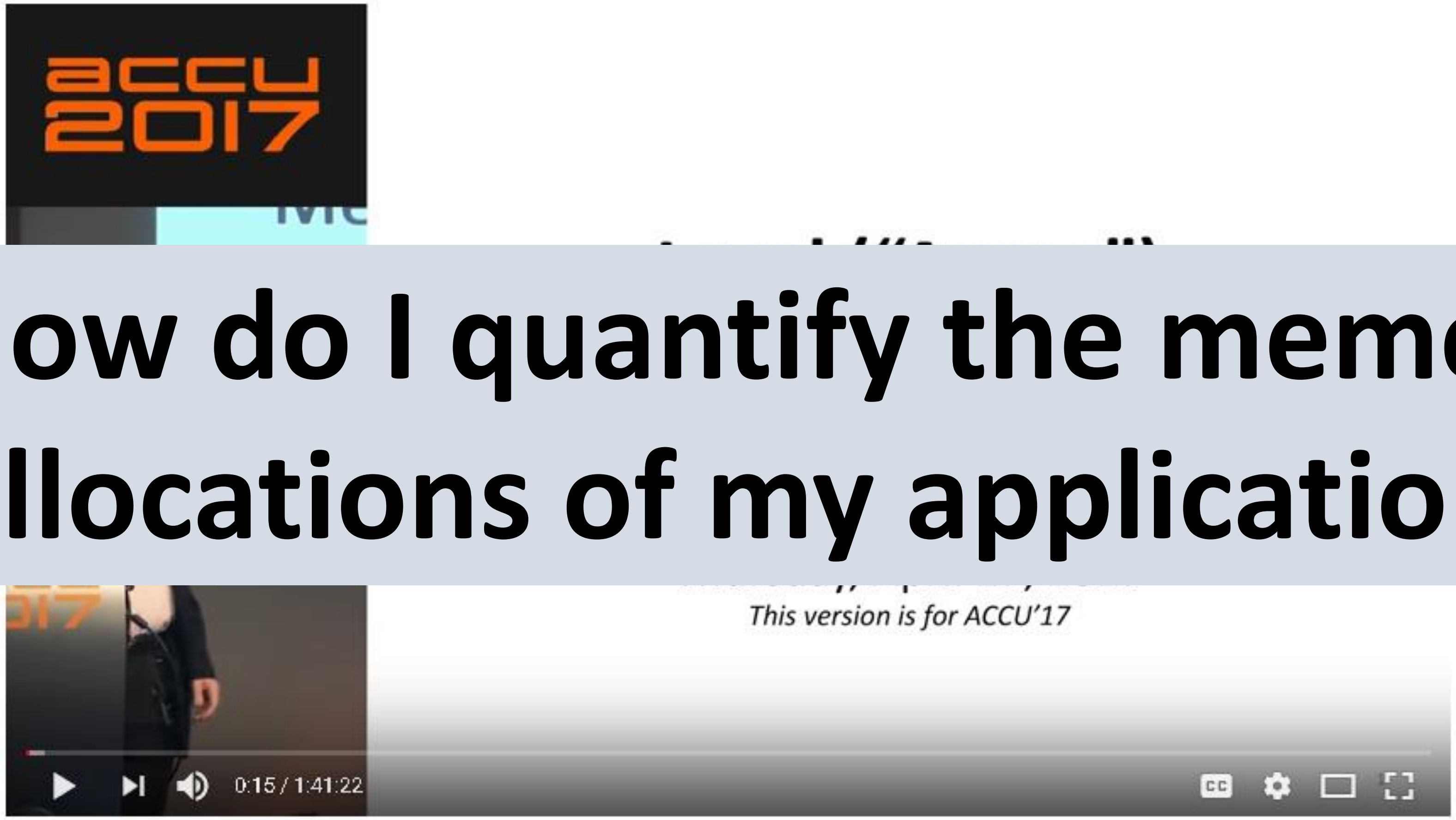
Custom allocators are a much discussed topic in the C++ industry.



Local (arena) Memory Allocators - John Lakos [ACCU 2017]

Extensive benchmarking: P01213R0, P0089R1

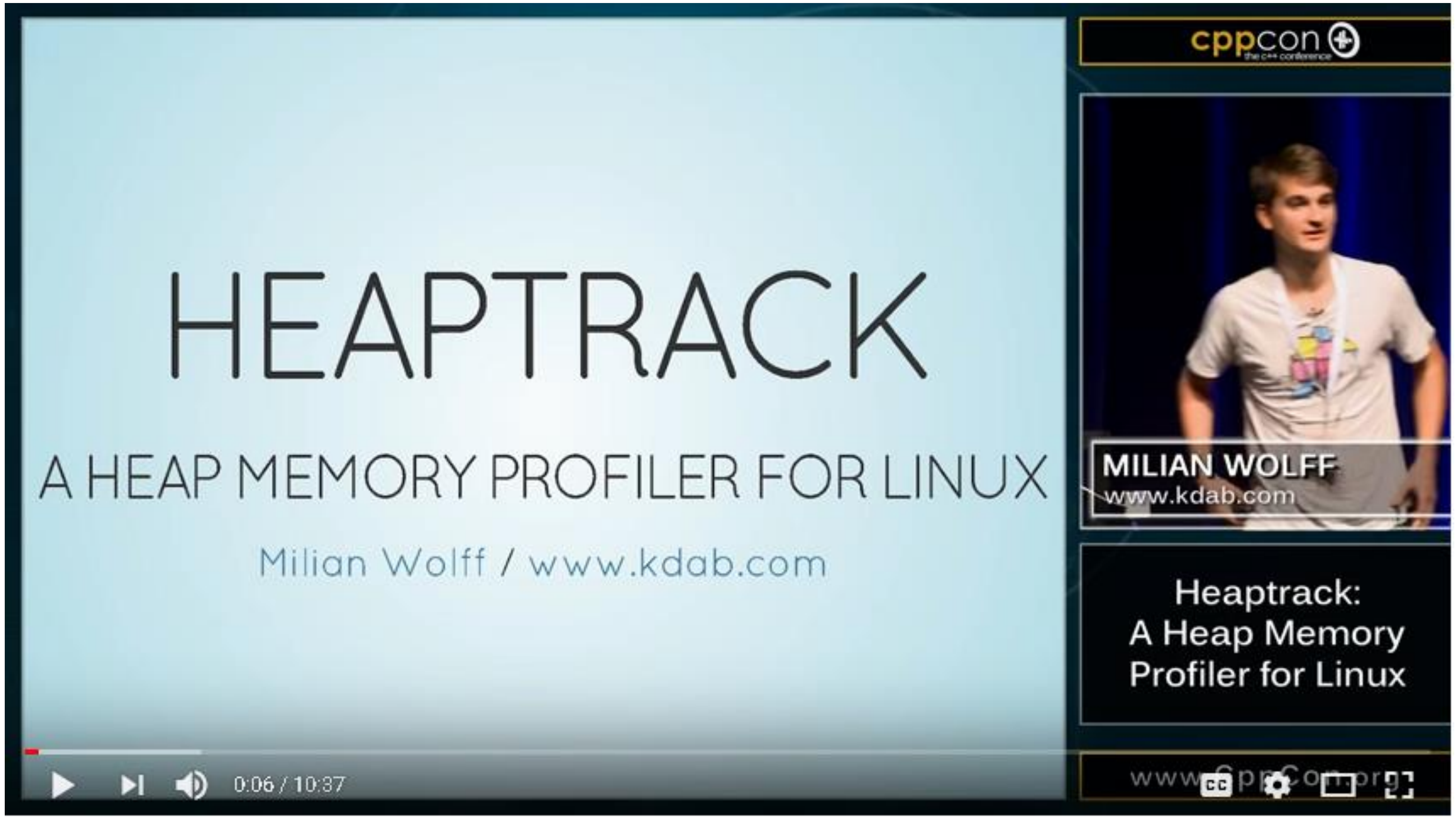
Custom allocators are a much discussed topic in the C++ industry.



How do I quantify the memory allocations of my application ?

Local (arena) Memory Allocators - John Lakos [ACCU 2017]

Extensive benchmarking: P01213R0, P0089R1

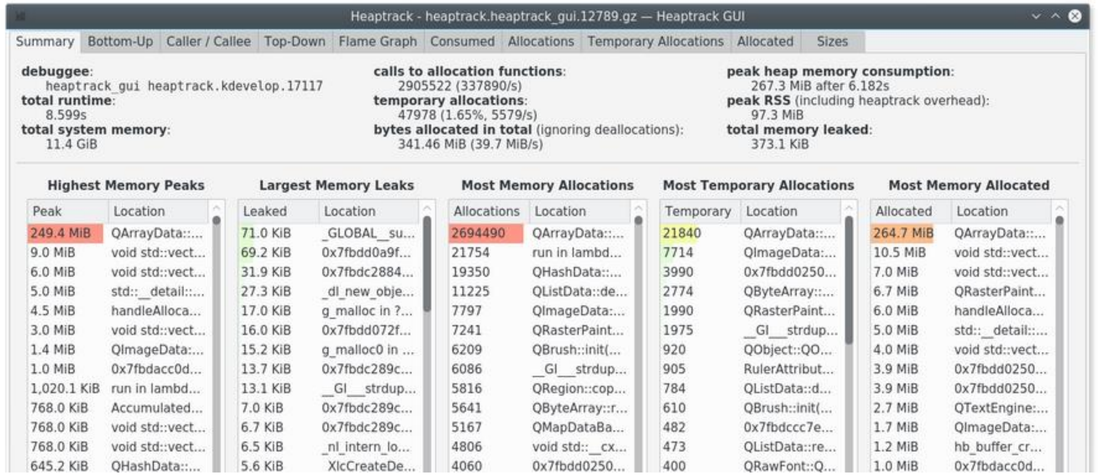

HEAPTRACK

A HEAP MEMORY PROFILER FOR LINUX

Milian Wolff / www.kdab.com

CppCon 2015: Milian Wolff "Heaptrack: A Heap Memory Profiler for Linux"

heaptrack



debuggee:
heaptrack_gui heaptrack.kdevelop.17117
total runtime:
8.599s
total system memory:
11.4 GiB

calls to allocation functions:
2905522 (337890/s)
temporary allocations:
47978 (1.65%, 5579/s)
bytes allocated in total (ignoring deallocations):
341.46 MiB (39.7 MiB/s)

peak heap memory consumption:
267.3 MiB after 6.182s
peak RSS (including heaptrack overhead):
97.3 MiB
total memory leaked:
373.1 KiB

Highest Memory Peaks

Peak	Location
249.4 MiB	QArrayData::...
9.0 MiB	void std::vect...
6.0 MiB	void std::vect...
5.0 MiB	std::__detail::...
4.5 MiB	handleAlloca...
3.0 MiB	void std::vect...
1.4 MiB	QImageData:...
1.0 MiB	0x7fbdacc0d...
1,020.1 KiB	run in lambd...
768.0 KiB	Accumulated...
768.0 KiB	void std::vect...
768.0 KiB	void std::vect...
645.2 KiB	QHashData::...

Largest Memory Leaks

Leaked	Location
71.0 KiB	_GLOBAL__su...
69.2 KiB	0x7fbdd0a9f...
31.9 KiB	0x7fbdc2884...
27.3 KiB	_dl_new_obje...
17.0 KiB	g_malloc in ?...
16.0 KiB	0x7fbdd072f...
15.2 KiB	g_malloc0 in ...
13.7 KiB	0x7fbdc289c...
13.1 KiB	__GI___strdup...
7.0 KiB	0x7fbdc289c...
6.7 KiB	0x7fbdc289c...
6.5 KiB	_nl_intern_lo...
5.6 KiB	XlcCreateDe...

Most Memory Allocations

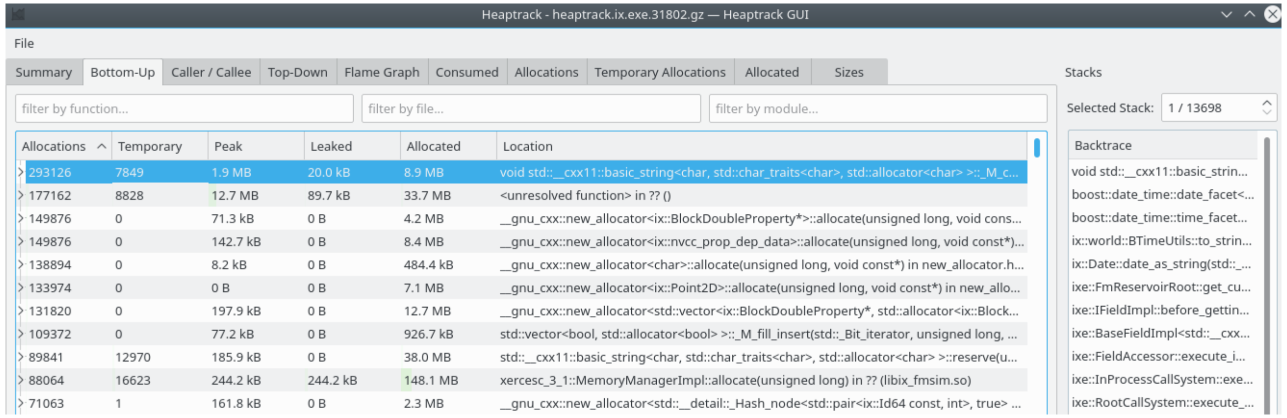
Allocations	Location
2694490	QArrayData::...
21754	run in lambd...
19350	QHashData::...
11225	QListData::de...
7797	QImageData:...
7241	QRasterPaint...
6209	QBrush::init(...
6086	__GI___strdup...
5816	QRegion::cop...
5641	QByteArray::r...
5167	QMapDataBa...
4806	void std::__cx...
4060	0x7fbdd0250...

Most Temporary Allocations

Temporary	Location
21840	QArrayData::...
7714	QImageData:...
3990	0x7fbdd0250...
2774	QByteArray::...
1990	QRasterPaint...
1975	__GI___strdup...
920	QObject::QO...
905	RulerAttribut...
784	QListData::d...
610	QBrush::init(...
482	0x7fbdccc7e...
473	QListData::re...
400	QRawFont::Q...

Most Memory Allocated

Allocated	Location
264.7 MiB	QArrayData::...
10.5 MiB	void std::vect...
7.0 MiB	void std::vect...
6.7 MiB	QRasterPaint...
6.0 MiB	handleAlloca...
5.0 MiB	std::__detail::...
4.0 MiB	void std::vect...
3.9 MiB	0x7fbdd0250...
3.9 MiB	0x7fbdd0250...
2.7 MiB	QTextEngine:...
1.7 MiB	QImageData:...
1.2 MiB	hb_buffer_cr...
1.0 MiB	0x7fbdacc0d...



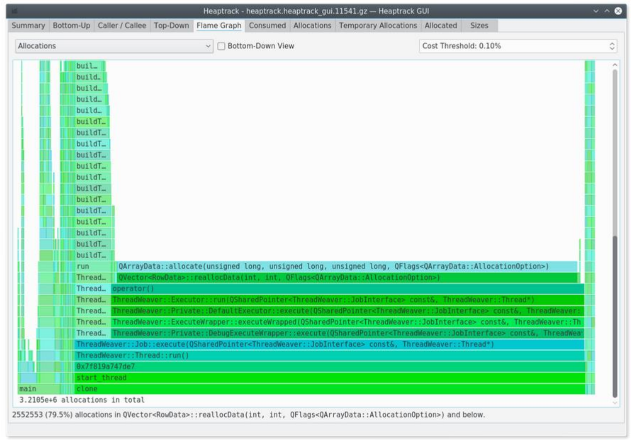
Allocations	Temporary	Peak	Leaked	Allocated	Location
293126	7849	1.9 MB	20.0 kB	8.9 MB	void std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::_M_c...
177162	8828	12.7 MB	89.7 kB	33.7 MB	<unresolved function> in ?? ()
149876	0	71.3 kB	0 B	4.2 MB	__gnu_cxx::new_allocator<ix::BlockDoubleProperty*>::allocate(unsigned long, void cons...
149876	0	142.7 kB	0 B	8.4 MB	__gnu_cxx::new_allocator<ix::nvcc_prop_dep_data>::allocate(unsigned long, void const*)...
138894	0	8.2 kB	0 B	484.4 kB	__gnu_cxx::new_allocator<char>::allocate(unsigned long, void const*) in new_allocator.h...
133974	0	0 B	0 B	7.1 MB	__gnu_cxx::new_allocator<ix::Point2D>::allocate(unsigned long, void const*) in new_allo...
131820	0	197.9 kB	0 B	12.7 MB	__gnu_cxx::new_allocator<std::vector<ix::BlockDoubleProperty*, std::allocator<ix::Block...
109372	0	77.2 kB	0 B	926.7 kB	std::vector<bool, std::allocator<bool> >::_M_fill_insert(std::_Bit_iterator, unsigned long, ...
89841	12970	185.9 kB	0 B	38.0 MB	std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::reserve(u...
88064	16623	244.2 kB	244.2 kB	148.1 MB	xercesc_3_1::MemoryManagerImpl::allocate(unsigned long) in ?? (libix_fmsim.so)
71063	1	161.8 kB	0 B	2.3 MB	__gnu_cxx::new_allocator<std::__detail::_Hash_node<std::pair<ix::Id64 const, int>, true> ...

Stacks

Selected Stack: 1 / 13698

Backtrace
void std::__cxx11::basic_strin...
boost::date_time::date_facet<...
boost::date_time::time_facet...
ix::world::BTimeUtils::to_strin...
ix::Date::date_as_string(std::_...
ixe::FmReservoirRoot::get_cu...
ixe::IFieldImpl::before_gettin...
ixe::BaseFieldImpl<std::__cxx...
ixe::FieldAccessor::execute_i...
ixe::InProcessCallSystem::exe...
ixe::RootCallSystem::execute_...

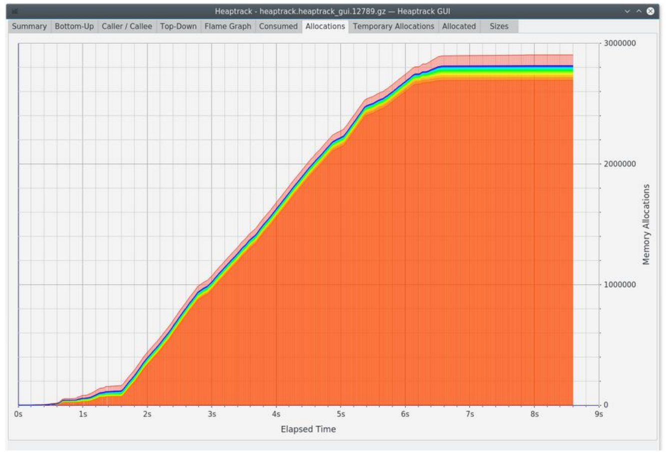
heaptrack



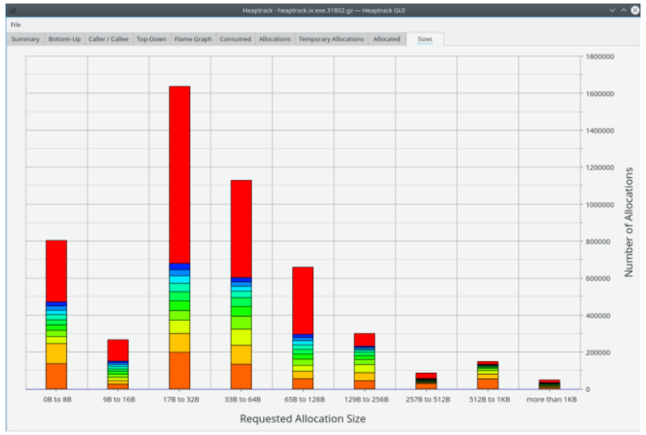
```
QArrayData::allocate(unsigned long, unsigned long, unsigned long, QFlags<QArrayData::AllocationOption>)
QVector<RowData>::reallocData(int, int, QFlags<QArrayData::AllocationOption>)
operator()
ThreadWeaver::Executor::run(QSharedPointer<ThreadWeaver::JobInterface> const&, ThreadWeaver::Thread*)
ThreadWeaver::Private::DefaultExecutor::execute(QSharedPointer<ThreadWeaver::JobInterface> const&, ThreadWeaver:
ThreadWeaver::ExecuteWrapper::executeWrapped(QSharedPointer<ThreadWeaver::JobInterface> const&, ThreadWeaver::Th
ThreadWeaver::Private::DebugExecuteWrapper::execute(QSharedPointer<ThreadWeaver::JobInterface> const&, ThreadWea
ThreadWeaver::Job::execute(QSharedPointer<ThreadWeaver::JobInterface> const&, ThreadWeaver::Thread*)
ThreadWeaver::Thread::run()
0x7f819a747de7
start_thread
main   clone
3.2105e+6 allocations in total
```

2552553 (79.5%) allocations in QVector<RowData>::reallocData(int, int, QFlags<QArrayData::AllocationOption>) and below.

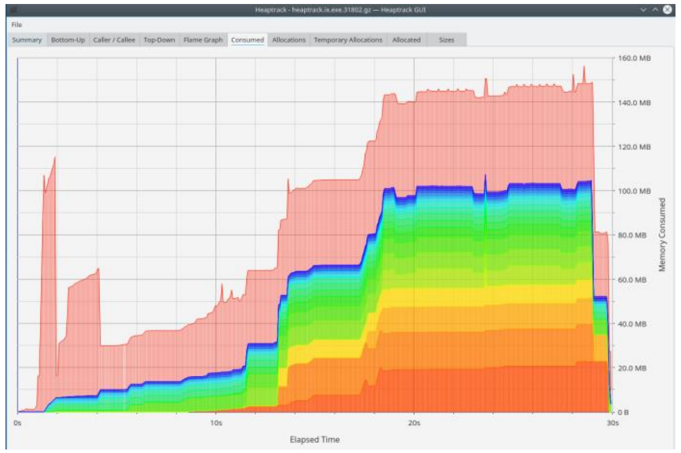
Flamecharts



Cumulated allocations

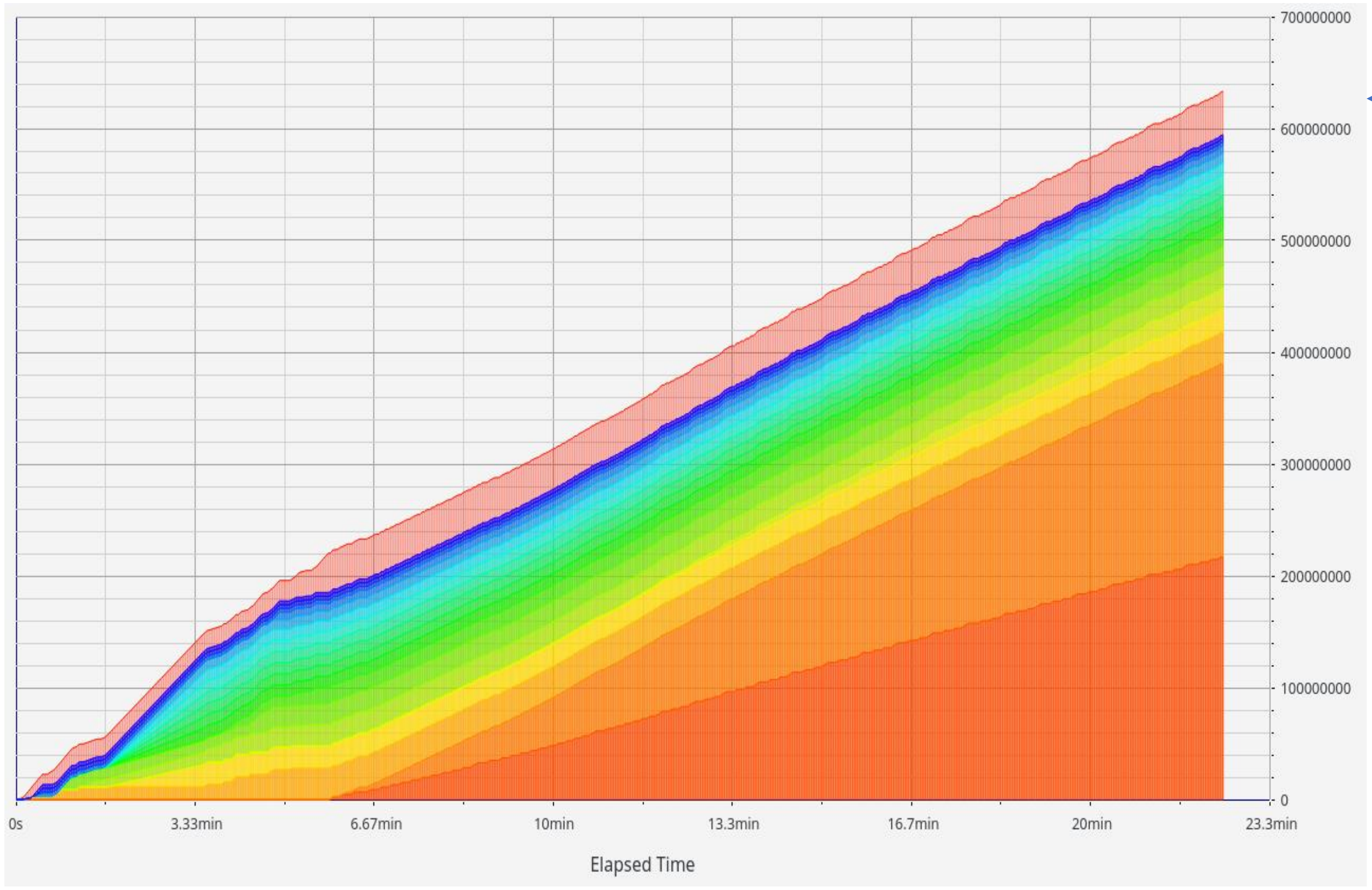


Sizes



Consumed

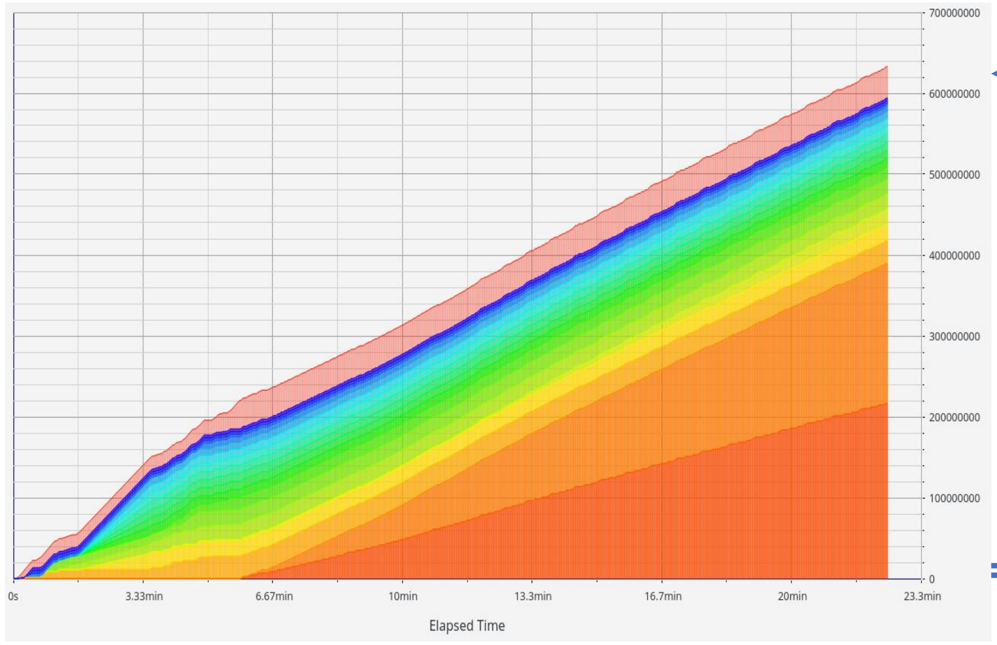
A case study



~630e6

Cumulated allocations

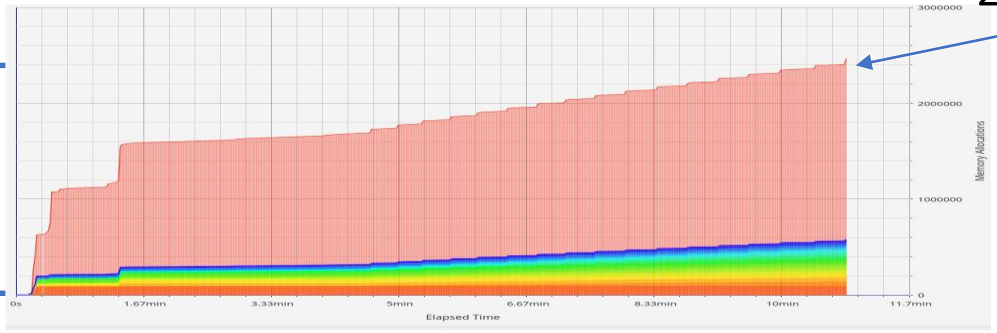
A case study



~630e6

Number of allocations reduced by x250

~2.5e6



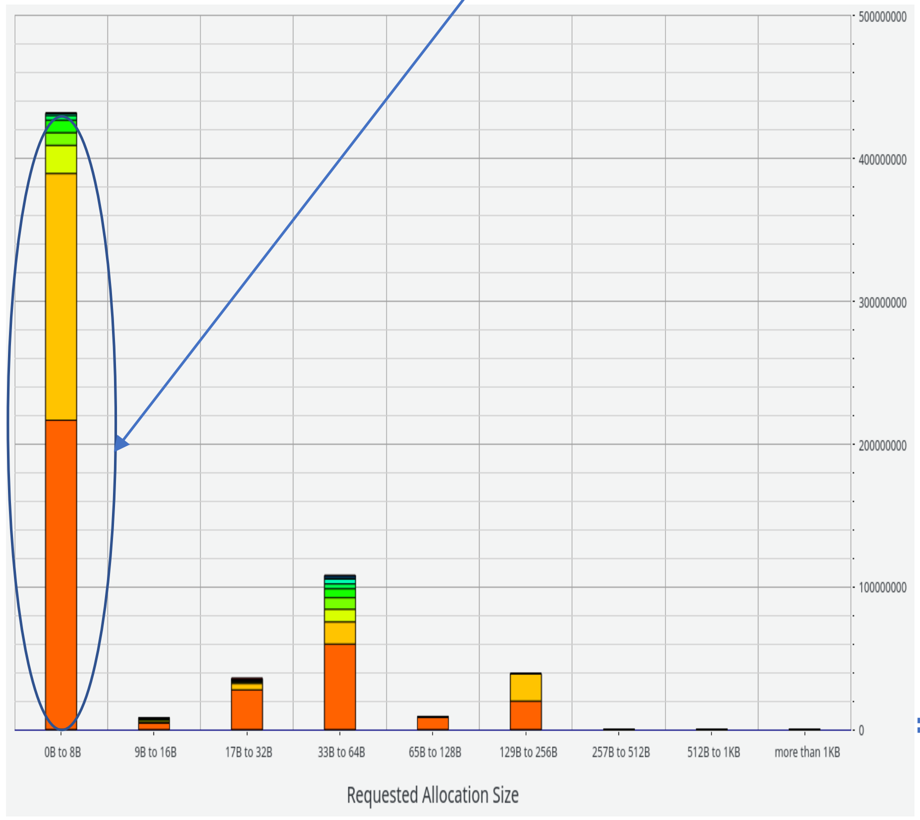
Before

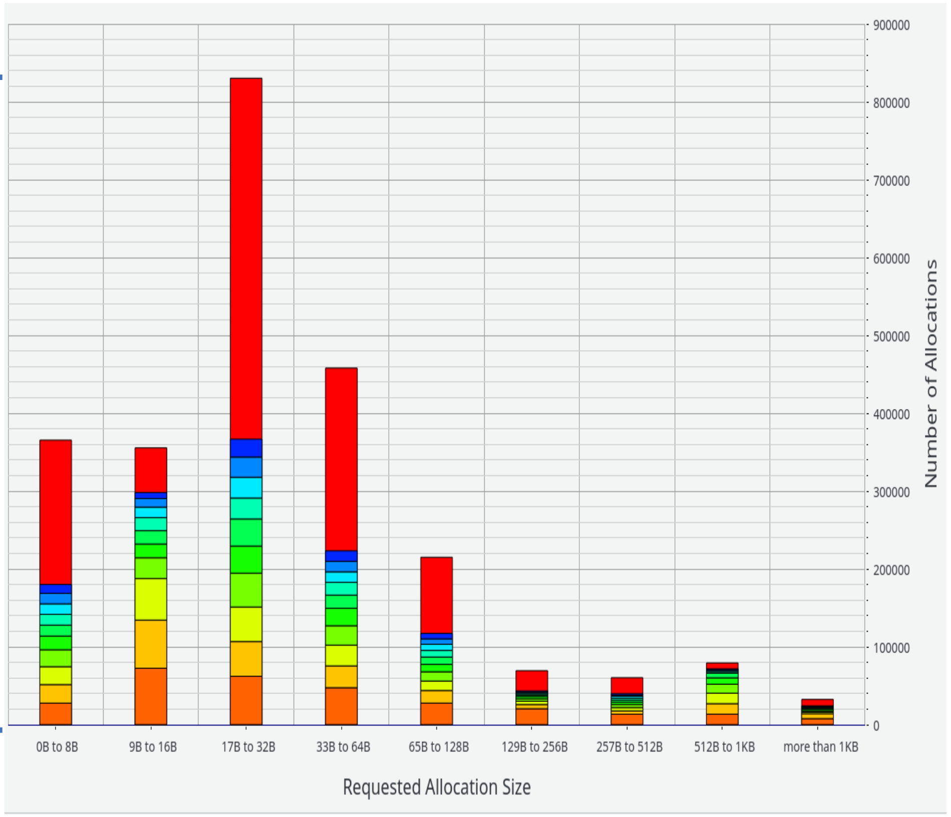
After

A case study

Most allocations are for 8 bytes or less.

450e6





0.8e6

Before

After

Solutions

- Do not copy if you can.
  - Avoid unused objects.
  - Use references.
  - Use views (`gsl::span`, `std::string_view`).
  - Use moves.
- Avoid allocation.
  - Use `std::array`, `boost::container::small_vector`
  - Avoid pimpl when necessary. Use `std::optional`.
- Re-use allocated memory.
  - Use `std::vector::reserve()`.
  - Make use of `std::vector` capacity.
- Use contiguous containers.
  - Avoid when possible `std::map`, `std::set` and `std::list` in critical code.
  - Use local memory allocator for node-based containers when appropriate

Solutions

- Do not copy if you can.
  - Avoid unused objects.
  - Use references.
  - Use views (`gsl::span`, `std::string_view`).
  - Use moves.
- Avoid allocation.
  - Use `std::array`, `boost::container::small_vector`
  - Avoid pimpl when necessary. Use `std::optional`.
- Re-use allocated memory.
  - Use `std::vector::reserve()`.
  - Make use of `std::vector` capacity.
- Use contiguous containers.
  - Avoid when possible `std::map`, `std::set` and `std::list` in critical code.
  - Use local memory allocator for node-based containers when appropriate

Solutions

- Do not copy if you can.
  - Avoid unused objects.
  - Use references.
  - Use views (`gsl::span`, `std::string_view`).
  - Use moves.
- Avoid allocation.
  - Use `std::array`, `boost::container::small_vector`
  - Avoid pimpl when necessary. Use `std::optional`.
- Re-use allocated memory.
  - Use `std::vector::reserve()`.
  - Make use of `std::vector` capacity.
- Use contiguous containers.
  - Avoid when possible `std::map`, `std::set` and `std::list` in critical code.
  - Use local memory allocator for node-based containers when appropriate

Solutions

- Do not copy if you can.
  - Avoid unused objects.
  - Use references.
  - Use views (`gsl::span`, `std::string_view`).
  - Use moves.
- Avoid allocation.
  - Use `std::array`, `boost::container::small_vector`
  - Avoid pimpl when necessary. Use `std::optional`.
- Re-use allocated memory.
  - Use `std::vector::reserve()`.
  - Make use of `std::vector` capacity.
- Use contiguous containers.
  - Avoid when possible `std::map`, `std::set` and `std::list` in critical code.
  - Use local memory allocator for node-based containers when appropriate

Lessons learned

Go to conferences !

    or watch them on YouTube.

Do not be afraid to ask questions.

    at conferences or on the web.

Try new tools.

    … and make improvements thanks to them.