Memory Management Needs of a Computational Algebra System

Steve Linton, St Andrews

GAP

Groups, Algorithms, Programming

- The mathematician's handle on symmetry
- Key objects in pure and applied mathematics
- Early adopters of computation in pure maths
- Groups of interest are often infinite, or very large indeed
  - Study the group by computing with just a (carefully selected) few of its elements

"There will be positively no internal alterations to be made even if we wish suddenly to switch from calculating the energy levels of the neon atom to the enumeration of groups of order 720."

Alan Turing (1945)

808017424794512875886459904961710757
005754368000000000

Groups, Algorithms, Programming

- Given a concise description of a group
  - generating permutations or matrices
  - finite presentation
- Calculate global properties of group:
  - size,
  - composition factors,
  - membership test
  - character table
- Search for elements of the group with special properties
  - "find an element that moves this to that"
  - find all the unipotent matrices in the group

Rubik's Cube Group:

- Generated by 5 permutations of 48 small squares
- Size = $2^{27}3^{14}5^{3}7^{2}11$
- Structure:
  $(2^{11}x3^{7}):(A_{8}xA_{11}):2$
- No element that just twists one corner

Groups, Algorithms, Programming

GAP History

- Development began in Aachen, mid-80s
- Neubüser, Schönert, others
- 1997, Neubüser retired
  - international project coordinated from St Andrews till 2005
  - coordination now shared with three other centres
- Free Software under GPL
- Widely used and extended

GAP Numbers

- 174K lines of C
- 450K lines of GAP in core system
  - 4000+ operations
  - 10000+ methods
- 1M lines of GAP in 92 contributed packages
- 100MB + of data libraries
- 1350 pages in reference manual
- over 1000 citations

GAP In Action

```
    gap> AvgOrder :=
>  g->Sum(ConjugacyClasses(g),
>  c-> Size(c)*Order(Representative(c)))/
>   Size(g);
function( g ) ... end
gap> AvgOrder(MathieuGroup(11));
53131/7920
gap> ForAny(AllSmallGroups([2..100]),
> g->IsInt(AvgOrder(g)));
false
```

- Qn: is there a non-trivial group whose elements have integer average order?
- Dynamically typed language
- Functions are first class objects
- generic operations like Size and ConjugacyClasses
- higher-order functions like Sum, ForAny
- Not functional, but global side-effects are rare
- single threaded

GAP Usage

GAP Usage

- Most GAP usage is interactive

GAP Usage

- Most GAP usage is interactive
- Extend library with functionality for your problem then solve problem interactively

GAP Usage

- Most GAP usage is interactive
- Extend library with functionality for your problem then solve problem interactively
  - advanced users might extend kernel too

GAP Usage

- Most GAP usage is interactive
- Extend library with functionality for your problem then solve problem interactively
  - advanced users might extend kernel too
- Some calculations run for weeks

GAP Usage

- Most GAP usage is interactive
- Extend library with functionality for your problem then solve problem interactively
    - advanced users might extend kernel too
- Some calculations run for weeks
- Available memory is often limiting factor

GAP Usage

- Most GAP usage is interactive
- Extend library with functionality for your problem then solve problem interactively
  - advanced users might extend kernel too
- Some calculations run for weeks
- Available memory is often limiting factor
  - Need to allow close to 4GB of workspace on 32 bit systems

GAP Usage

- Most GAP usage is interactive
- Extend library with functionality for your problem then solve problem interactively
  - advanced users might extend kernel too
- Some calculations run for weeks
- Available memory is often limiting factor
  - Need to allow close to 4GB of workspace on 32 bit systems
  - Need to allow workspace as large as physical memory

GAP Usage

- Most GAP usage is interactive
- Extend library with functionality for your problem then solve problem interactively
  - advanced users might extend kernel too
- Some calculations run for weeks
- Available memory is often limiting factor
  - Need to allow close to 4GB of workspace on 32 bit systems
  - Need to allow workspace as large as physical memory
  - Might be handful of huge objects, might be hundreds of millions of tiny ones

GASMAN
the GAp Storage Manager

GASMAN
the GAp Storage Manager

- Written by Martin Schönert as part of a rewrite of GAP in mid-90s

GASMAN the GAp Storage Manager

- Written by Martin Schönert as part of a rewrite of GAP in mid-90s
- Two generation conservative mark & sweep GC.

GASMAN
the GAp Storage Manager

- Written by Martin Schönert as part of a rewrite of GAP in mid-90s
- Two generation conservative mark & sweep GC.
  - objects references are pointers to "master pointers" which point to actual objects

GASMAN
the GAp Storage Manager

- Written by Martin Schönert as part of a rewrite of GAP in mid-90s
- Two generation conservative mark & sweep GC.
  - objects references are pointers to “master pointers” which point to actual objects
    - master pointers never move, objects do

GASMAN
the GAp Storage Manager

- Written by Martin Schönert as part of a rewrite of GAP in mid-90s
- Two generation conservative mark & sweep GC.
  - objects references are pointers to “master pointers” which point to actual objects
    - master pointers never move, objects do
    - all master pointers in one contiguous area

GASMAN
the GAp Storage Manager

- Written by Martin Schönert as part of a rewrite of GAP in mid-90s
- Two generation conservative mark & sweep GC.
  - objects references are pointers to “master pointers” which point to actual objects
    - master pointers never move, objects do
    - all master pointers in one contiguous area
      - makes it easy to recognise object references

GASMAN
the GAp Storage Manager

- Written by Martin Schönert as part of a rewrite of GAP in mid-90s
- Two generation conservative mark & sweep GC.
  - objects references are pointers to “master pointers” which point to actual objects
    - master pointers never move, objects do
    - all master pointers in one contiguous area
      - makes it easy to recognise object references
  - objects have a header giving size and low-level type.

GASMAN
the GAp Storage Manager

- Written by Martin Schönert as part of a rewrite of GAP in mid-90s
- Two generation conservative mark & sweep GC.
  - objects references are pointers to "master pointers" which point to actual objects
    - master pointers never move, objects do
    - all master pointers in one contiguous area
      - makes it easy to recognise object references
  - objects have a header giving size and low-level type.
    - type determines which parts of the object need to be scanned for references

GASMAN
the GAp Storage Manager

- Written by Martin Schönert as part of a rewrite of GAP in mid-90s
- Two generation conservative mark & sweep GC.
  - objects references are pointers to "master pointers" which point to actual objects
    - master pointers never move, objects do
    - all master pointers in one contiguous area
      - makes it easy to recognise object references
  - objects have a header giving size and low-level type.
    - type determines which parts of the object need to be scanned for references
- Object references can also be "fake" pointers encoding small integers or finite field elements

GASMAN Rules for kernel programming

GASMAN Rules for kernel programming

- global and static variables that might contain object references must be declared to GASMAN

GASMAN Rules for kernel programming

- global and static variables that might contain object references must be declared to GASMAN
    - local variables are picked up in a sweep of the stack

GASMAN Rules for kernel programming

- global and static variables that might contain object references must be declared to GASMAN
    - local variables are picked up in a sweep of the stack
- Mustn't keep an actual pointer into an object across a potential garbage collection

GASMAN Rules for kernel programming

- global and static variables that might contain object references must be declared to GASMAN
  - local variables are picked up in a sweep of the stack
- Mustn't keep an actual pointer into an object across a potential garbage collection
  - references are fine

GASMAN Rules for kernel programming

- global and static variables that might contain object references must be declared to GASMAN
  - local variables are picked up in a sweep of the stack
- Mustn’t keep an actual pointer into an object across a potential garbage collection
  - references are fine
- Call CHANGED_BAG whenever you (might) add a new object reference to an old object.

GASMAN Rules for kernel programming

- global and static variables that might contain object references must be declared to GASMAN
    - local variables are picked up in a sweep of the stack
- Mustn't keep an actual pointer into an object across a potential garbage collection
    - references are fine
- Call CHANGED_BAG whenever you (might) add a new object reference to an old object.
- Don't use malloc too much.

GASMAN Rules for kernel programming

- global and static variables that might contain object references must be declared to GASMAN
    - local variables are picked up in a sweep of the stack
- Mustn't keep an actual pointer into an object across a potential garbage collection
    - references are fine
- Call CHANGED_BAG whenever you (might) add a new object reference to an old object.
- Don't use malloc too much.
    - GASMAN uses sbrk and likes a contiguous workspace

A Few More Features

A Few More Features

- Save & load workspace -- number all the objects, store references as numbers

A Few More Features

- Save & load workspace -- number all the objects, store references as numbers
    - saved workspaces not safe across endianness or wordsize changes, or any change to the system.

A Few More Features

- Save & load workspace -- number all the objects, store references as numbers
  - saved workspaces not safe across endianness or wordsize changes, or any change to the system.
- Weak pointers -- not used a lot, but valuable where they are used

Pros and Cons

Pros and Cons

- we have it, it works

Pros and Cons

- we have it, it works
- highly tuned for our needs

Pros and Cons

- we have it, it works
- highly tuned for our needs
- highly portable

Pros and Cons

- we have it, it works
- highly tuned for our needs
- highly portable
  - UNIX, cygwin, MacOS

Pros and Cons

- we have it, it works
- highly tuned for our needs
- highly portable
  - UNIX, cygwin, MacOS
  - No dependency on OS/hardware features (almost)

Pros and Cons

- we have it, it works
- highly tuned for our needs
- highly portable
  - UNIX, cygwin, MacOS
  - No dependency on OS/hardware features (almost)

Pros and Cons

- we have it, it works
- highly tuned for our needs
- highly portable
  - UNIX, cygwin, MacOS
  - No dependency on OS/hardware features (almost)

- a bit more time spent in GC than we'd really like

Pros and Cons

- we have it, it works
- highly tuned for our needs
- highly portable
  - UNIX, cygwin, MacOS
  - No dependency on OS/hardware features (almost)

- a bit more time spent in GC than we'd really like
- Not friendly to FOS libraries

Pros and Cons

- we have it, it works
- highly tuned for our needs
- highly portable
  - UNIX, cygwin, MacOS
  - No dependency on OS/hardware features (almost)

- a bit more time spent in GC than we'd really like
- Not friendly to FOS libraries
- we have to maintain it

Pros and Cons

- we have it, it works
- highly tuned for our needs
- highly portable
  - UNIX, cygwin, MacOS
  - No dependency on OS/hardware features (almost)

- a bit more time spent in GC than we'd really like
- Not friendly to FOS libraries
- we have to maintain it
- Cost of double indirections

Pros and Cons

- we have it, it works
- highly tuned for our needs
- highly portable
  - UNIX, cygwin, MacOS
  - No dependency on OS/hardware features (almost)

- a bit more time spent in GC than we'd really like
- Not friendly to FOS libraries
- we have to maintain it
- Cost of double indirections
- Awkward kernel programming discipline

Pros and Cons

- we have it, it works
- highly tuned for our needs
- highly portable
  - UNIX, cygwin, MacOS
  - No dependency on OS/hardware features (almost)

- a bit more time spent in GC than we'd really like
- Not friendly to FOS libraries
- we have to maintain it
- Cost of double indirections
- Awkward kernel programming discipline
- Not thread-safe!

HPC-GAP

HPC-GAP

- EPSRC project 2009-2013

HPC-GAP

- EPSRC project 2009-2013
- Deliver a version of GAP capable of exploiting full range of parallel hardware

HPC-GAP

- EPSRC project 2009-2013
- Deliver a version of GAP capable of exploiting full range of parallel hardware
- From multicore to supercomputers

HPC-GAP

- EPSRC project 2009-2013
- Deliver a version of GAP capable of exploiting full range of parallel hardware
- From multicore to supercomputers
- Now have multi-threaded GAP

HPC-GAP

- EPSRC project 2009-2013
- Deliver a version of GAP capable of exploiting full range of parallel hardware
- From multicore to supercomputers
- Now have multi-threaded GAP
- Temporarily replaced GASMAN by Boehm GC

HPC-GAP

- EPSRC project 2009-2013
- Deliver a version of GAP capable of exploiting full range of parallel hardware
- From multicore to supercomputers
- Now have multi-threaded GAP
- Temporarily replaced GASMAN by Boehm GC
  - significant performance loss -- c 15%

HPC-GAP

- EPSRC project 2009-2013
- Deliver a version of GAP capable of exploiting full range of parallel hardware
- From multicore to supercomputers
- Now have multi-threaded GAP
- Temporarily replaced GASMAN by Boehm GC
  - significant performance loss -- c 15%
  - and some bugs on 64 bit.

Dataspaces

Dataspaces

- Introduced in HPCGAP to make it easier to write thread-safe programs

Dataspaces

- Introduced in HPCGAP to make it easier to write thread-safe programs
    - we hope

Dataspaces

- Introduced in HPCGAP to make it easier to write thread-safe programs
  - we hope
- Three kinds:

Dataspaces

- Introduced in HPCGAP to make it easier to write thread-safe programs
  - we hope
- Three kinds:
  - thread-local -- one of these associated with each thread

Dataspaces

- Introduced in HPCGAP to make it easier to write thread-safe programs
    - we hope
- Three kinds:
    - thread-local -- one of these associated with each thread
    - global -- one of these

Dataspaces

- Introduced in HPCGAP to make it easier to write thread-safe programs
  - we hope
- Three kinds:
  - thread-local -- one of these associated with each thread
  - global -- one of these
  - shared -- possibly many of these, typically each with just a few objects

Dataspaces

- Introduced in HPCGAP to make it easier to write thread-safe programs
  - we hope
- Three kinds:
  - thread-local -- one of these associated with each thread
  - global -- one of these
  - shared -- possibly many of these, typically each with just a few objects
- Newly created mutable objects in thread-local dataspace

Dataspaces

- Introduced in HPCGAP to make it easier to write thread-safe programs
  - we hope
- Three kinds:
  - thread-local -- one of these associated with each thread
  - global -- one of these
  - shared -- possibly many of these, typically each with just a few objects
- Newly created mutable objects in thread-local dataspace
  - Can we use this for GC?

What would we like?

What would we like?

- Get back at least the performance we had before

What would we like?

- Get back at least the performance we had before
- Thread-safe

What would we like?

- Get back at least the performance we had before
- Thread-safe
- Tolerant of malloc

What would we like?

- Get back at least the performance we had before
- Thread-safe
- Tolerant of malloc
- Allows at least some objects not to move around

What would we like?

- Get back at least the performance we had before
- Thread-safe
- Tolerant of malloc
- Allows at least some objects not to move around
- Memory efficient

What would we like?

- Get back at least the performance we had before
- Thread-safe
- Tolerant of malloc
- Allows at least some objects not to move around
- Memory efficient
- Still portable and reasonably easy to build

What would we like?

- Get back at least the performance we had before
- Thread-safe
- Tolerant of malloc
- Allows at least some objects not to move around
- Memory efficient
- Still portable and reasonably easy to build
- Some way to handle smallints etc. efficiently

What would we like?

- Get back at least the performance we had before
- Thread-safe
- Tolerant of malloc
- Allows at least some objects not to move around
- Memory efficient
- Still portable and reasonably easy to build
- Some way to handle smallints etc. efficiently
- Will scale to TB sized workspaces

Some Numbers

Some Numbers

- Starting Workspace: 375K objects, 30.5MB

Some Numbers

- Starting Workspace: 375K objects, 30.5MB
- First test to GC spontaneously in 70MB:

Some Numbers

- Starting Workspace: 375K objects, 30.5MB
- First test to GC spontaneously in 70MB:
  - young generation: 8429 objects/0.5MB survive, 593K objects/15MB dead

Some Behaviour

```
#G  PART    2613/   3630kb+live   142728/   3986kb+dead     8799/  71680kb free
#G  PART       5/   8000kb+live        0/      0kb+dead  4194103/  71680kb free
#G  FULL  504940/  50604kb live   844139/  34697kb dead     8550/  76288kb free
#G  PART       3/   8000kb+live        3/      0kb+dead  4192854/  76288kb free
#G  FULL  504938/  54604kb live        5/      0kb dead     9718/  83456kb free
#G  PART       3/  12000kb+live        0/      0kb+dead  4192022/  83456kb free
#G  FULL  504938/  60604kb live        0/      0kb dead    10374/  90112kb free
#G  PART       2/  12000kb+live        3/      0kb+dead  4188678/  90112kb free
#G  FULL  504938/  66604kb live        3/      0kb dead G12 8RZ.11638/ 101376kb free
#G  PART       2/  16000kb+live        0/      0kb+dead  4189942/ 101376kb free
#G  FULL  504938/  74604kb live        0/      0kb dead    12854/ 110592kb free
#G  PART       2/  16000kb+live        9/      0kb+dead  4183158/ 110592kb free
#G  FULL  504938/  82604kb live        9/      0kb dead    15286/ 129024kb free
#G  PART       1/  16000kb+live        0/      0kb+dead  4193590/ 129024kb free
#G  FULL  504938/  90604kb live        0/      0kb dead    16502/ 138240kb free
#G  PART       2/  32000kb+live        0/      0kb+dead  4178806/ 138240kb free
#G  FULL  504938/ 106604kb live        0/      0kb dead    18422/ 156160kb free
#G  PART      43/  33125kb+live      495/      8kb+dead     1045/ 156160kb free
#G  FULL  504941/  43729kb live      533/  80009kb dead    13079/  72192kb free
```

Conclusion

Conclusion

- We need a new Garbage Collector

Conclusion

- We need a new Garbage Collector
- We'd rather not write it

Conclusion

- We need a new Garbage Collector
- We'd rather not write it
- We have lots of varied real workloads

Conclusion

- We need a new Garbage Collector
- We'd rather not write it
- We have lots of varied real workloads
- We are interested in research cooperation, or in being pointed to a good GPL GC

Conclusion

- We need a new Garbage Collector
- We'd rather not write it
- We have lots of varied real workloads
- We are interested in research cooperation, or in being pointed to a good GPL GC
- Can you help?

Distributed Memory

- Also building infrastructure for distributed memory computing in GAP
- Building higher-level skeletons and data structured on top of MPI
- Some data structures might need some form of GC?