Visualisation of the Boehm-Demers-Weiser Conservative Garbage Collector

4th Year Project — 2001/02

Paul Dempster — 9805441

Department of Computer Science

University of Glasgow

Garbage Collection

- Automated memory management
  - Remove errors
  - Reduce development time
  - Increase performance?
- Everyone is using it then...
  - “I can do better”
  - Complex collector behaviour

Boehm-Demers-Weiser GC

- C based conservative collector
- 40,000 lines of code
- Multi-platform
- Userland support
- Widely used to provide GC to language runtimes

BDW Operation

- Conservative GC
- Mark & Sweep algorithm
- No separate GC thread
- Heap segmented into chunks and blocks
- Large and small objects
- Sweeping on demand
    - Small object blocks swept to satisfy allocation

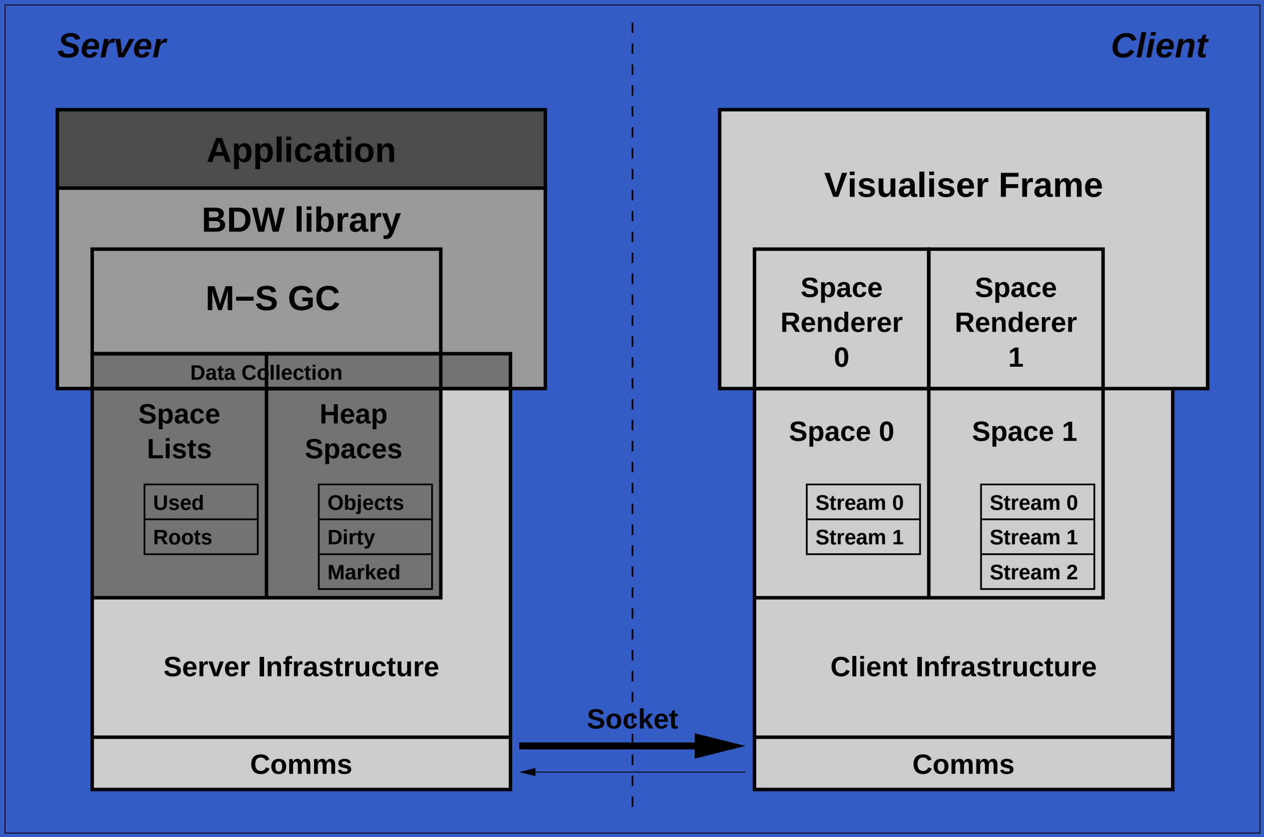
GCspy

- Generic heap visualisation framework
- Client-Server architecture
- Coarse-grain monitoring
- Presents attributes that implementor considers useful
- All customisation within server

Motivation

- No existing conservative collectors with GCspy support
- Test generic visualisation claim
- Provide insight into BDW GC operation
- Automatic support for many languages

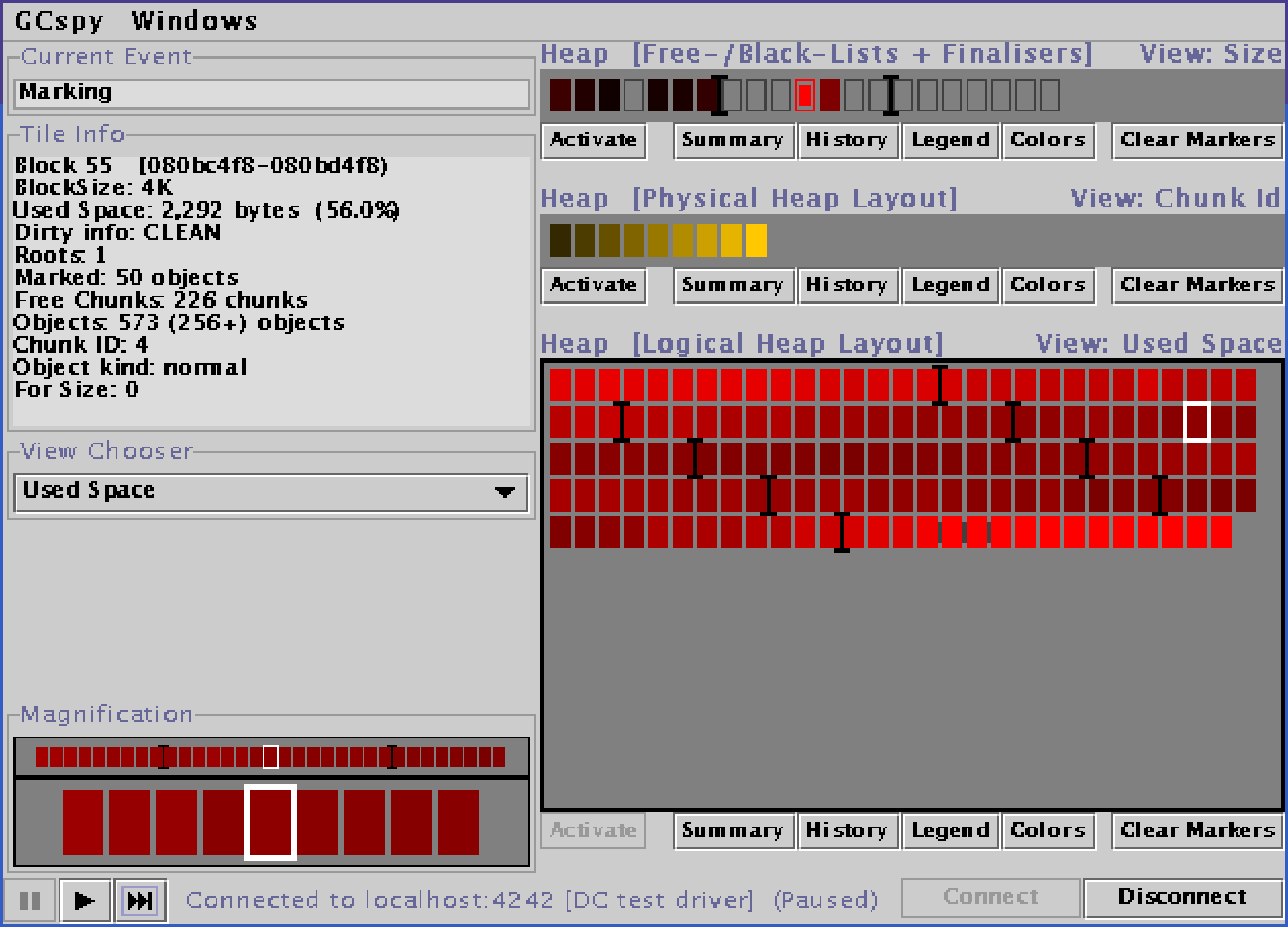
GCspy Architecture



- ⬜ Generic GCspy framework
- ▩ BDW GC library
- ▨ GC−dependent GCspy customisation
- ⬛ Running application

The Driver

- Maps collector state to GCspy abstractions
- Decides the shape of the visualisation
- Selected 3-Space design:
  - Main area shows block-level detail
  - 2nd area summaries previous at chunk level
  - Free-/Black-list and Finalisers area



Current Event: Marking

Tile Info

Block 55 [080bc4f8-080bd4f8)
BlockSize: 4K
Used Space: 2,292 bytes (56.0%)
Dirty info: CLEAN
Roots: 1
Marked: 50 objects
Free Chunks: 226 chunks
Objects: 573 (256+) objects
Chunk ID: 4
Object kind: normal
For Size: 0

View Chooser: Used Space

Magnification

Heap [Free-/Black-Lists + Finalisers] View: Size

Heap [Physical Heap Layout] View: Chunk Id

Heap [Logical Heap Layout] View: Used Space

Connected to localhost:4242 [DC test driver] (Paused)

Driver Structure

- Block data held per chunk; chunks held in a linked-list
- Secondary spaces data automatically calculated
- Generic enough to replace Mark&Sweep, Mark&Compact, etc. drivers
- GCspy framework required modification for expanding heaps.

Driver Tests

- Wrote an application which randomly allocated, and removed references to, objects
- Revealed no instabilities
- GCspy visualisation did reveal a bug in the test application!

Into the BDW GC

- Most time consuming part of the project
- Had to identify data structures that provide information we wish to visualise
- Code comments aimed at those already familiar with the collector
- Utilised “Understanding for C++” reverse-engineering software (Scientific Toolworks, Inc.)

Large Objects

- Caused a number of problems
- Supplied utility macros caused errors
    - Modified collecting strategy to treat them similarly to small objects
- Large objects crossing chunk boundaries unexpected
    - Added support to driver

Problems with Integration

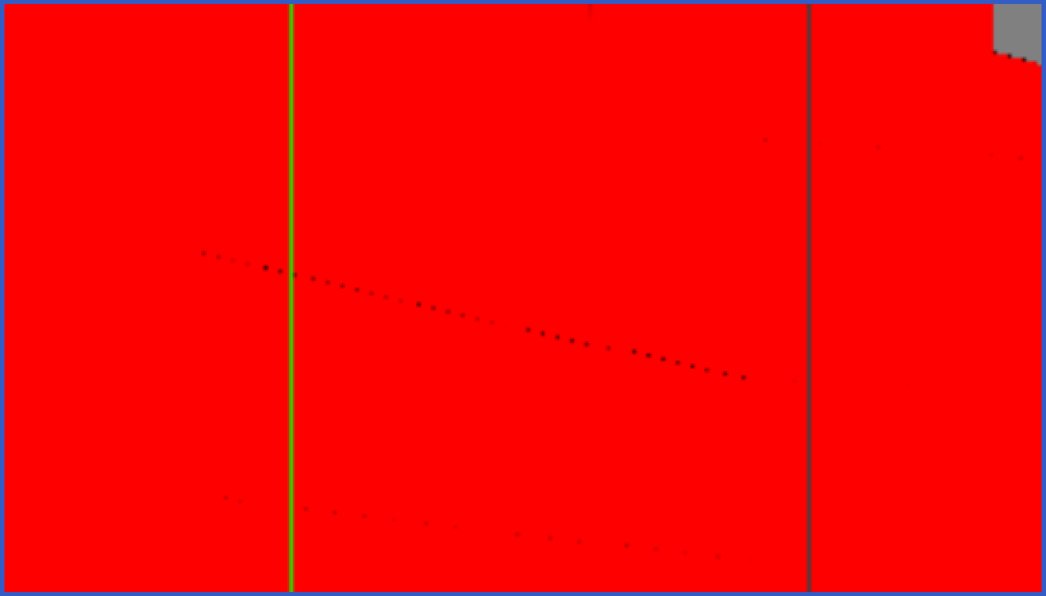
- Using a debugger difficult
  - Dirty bit mechanism stopped debugger at every line
- How to obtain roots data unclear
  - Strong suspicion driver is using the wrong data structure

Server Testing

- Tested integration with `gctest`, the collectors stress-test application
- Showed that GCspy code in the collector was stable and reliable.
    - Important for encouraging adoption
- Revealed interesting collector behaviour...

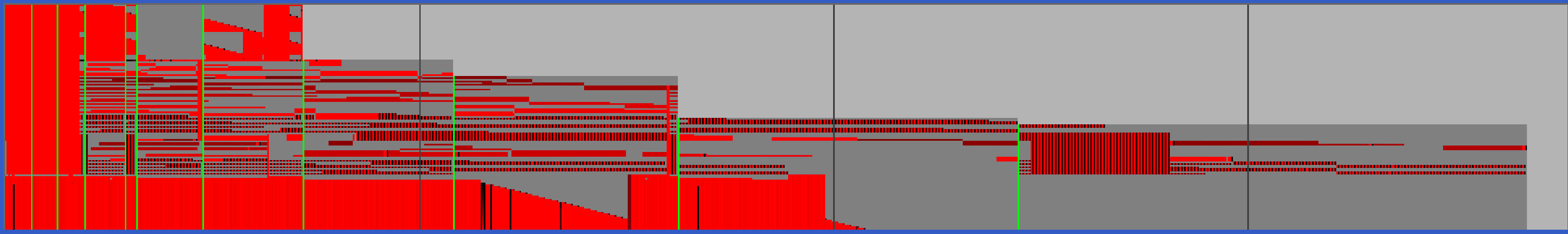
gctest

- Small object block sweeping could be seen in action



gctest

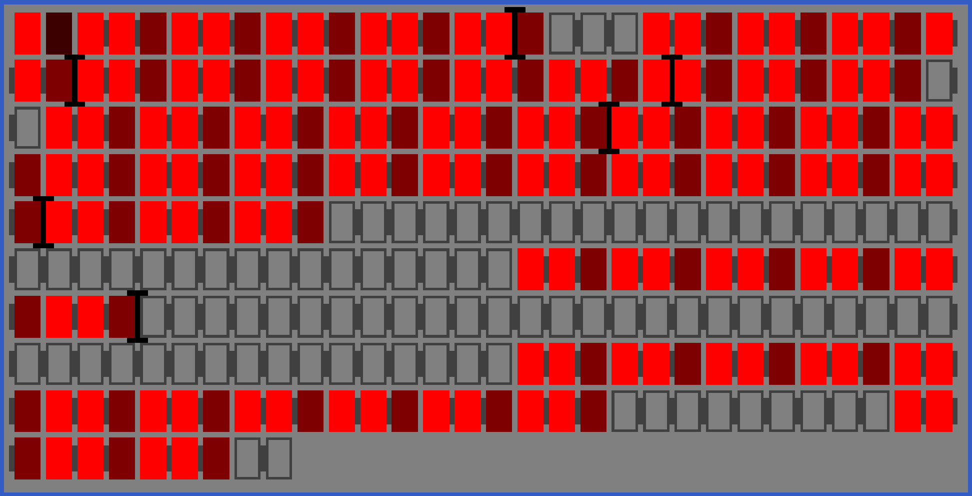
- The internal behaviour of the collector is shown



- Because GCspy is built into the collector, we can attribute this behaviour correctly

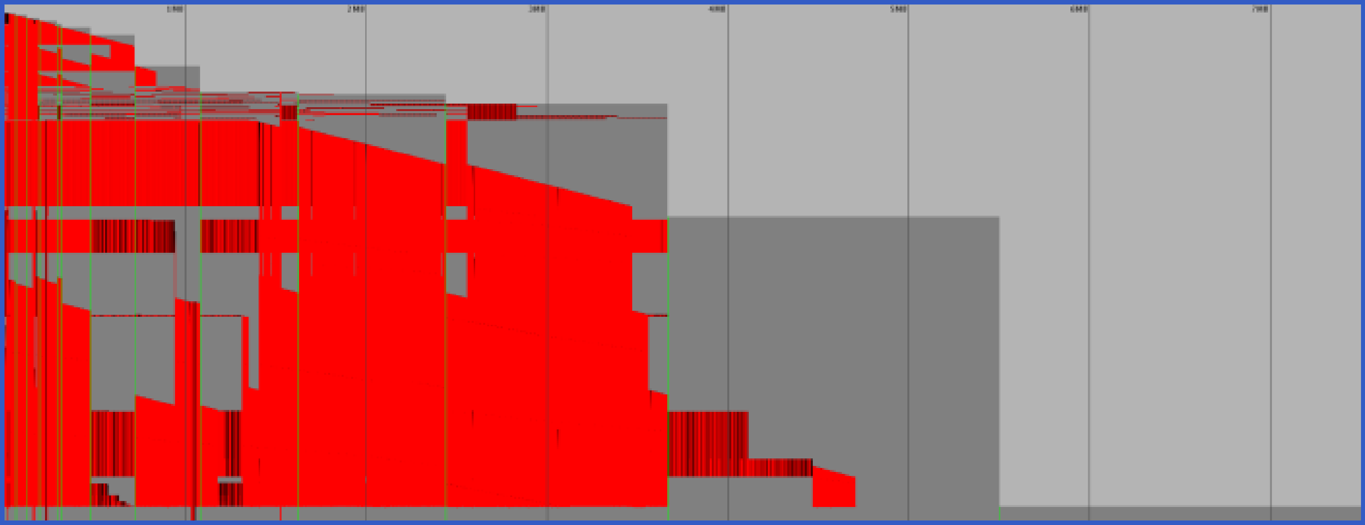
Large objects in Applications

- Visual patterns make it easy to identify when space is being wasted



Aggressive heap expansion

- For small applications the collector expands the heap too soon, and in too great increments



BDW GC Conclusions

- 14 years of development, works pretty well!
- GCspy reveals possible over-aggressive heap expansion
- Provides visual reference of expected behaviour for ports to other architectures
- Easy to distribute evidence of unusual behaviour

GCspy Conclusions

- Not quite generic enough
  - Required modifications make it even more flexible
- Highlights limitations in viewing a single stream at any instant
- Overall provides useful insight into the memory behaviour of collector and applications

Overall

- Provides GCspy support for widely used garbage collector
- First conservative collector supports generality claim
- BDW GC usage in other language runtimes provides wide potential userbase, particularly academic
- Allows programmers to see collectors really do know what they are doing