University of Glasgow

Poly/ML Heap Sizing

Jeremy.Singer@glasgow.ac.uk
David.Matthews@prolingua.co.uk

@jsinger_compsci


aimeric@o

Home

FAQ

Support

Documentation

Get Poly/ML

Poly/ML

This is the Poly/ML home page.

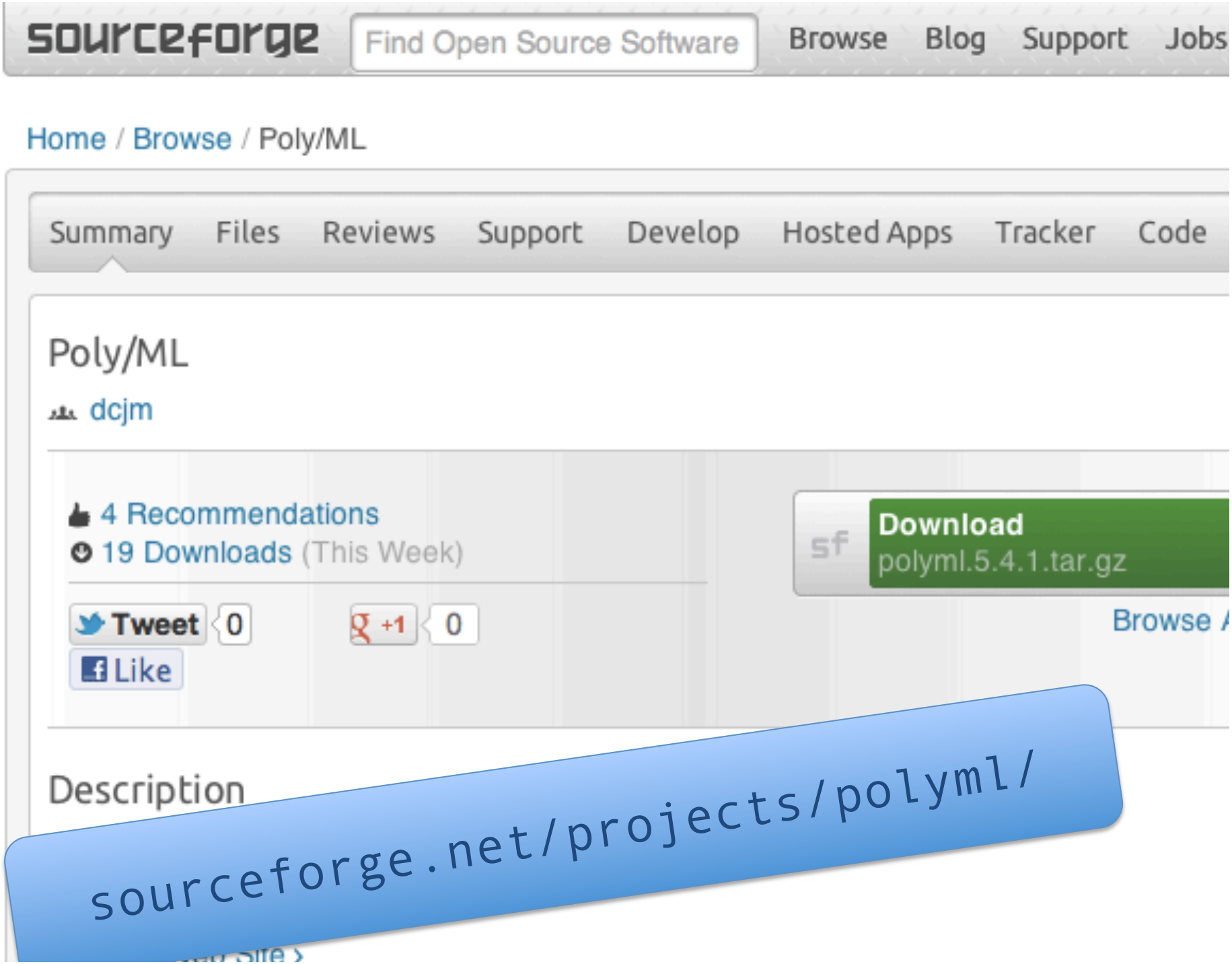
Poly/ML is a full implementation of Standard ML available as open-source.

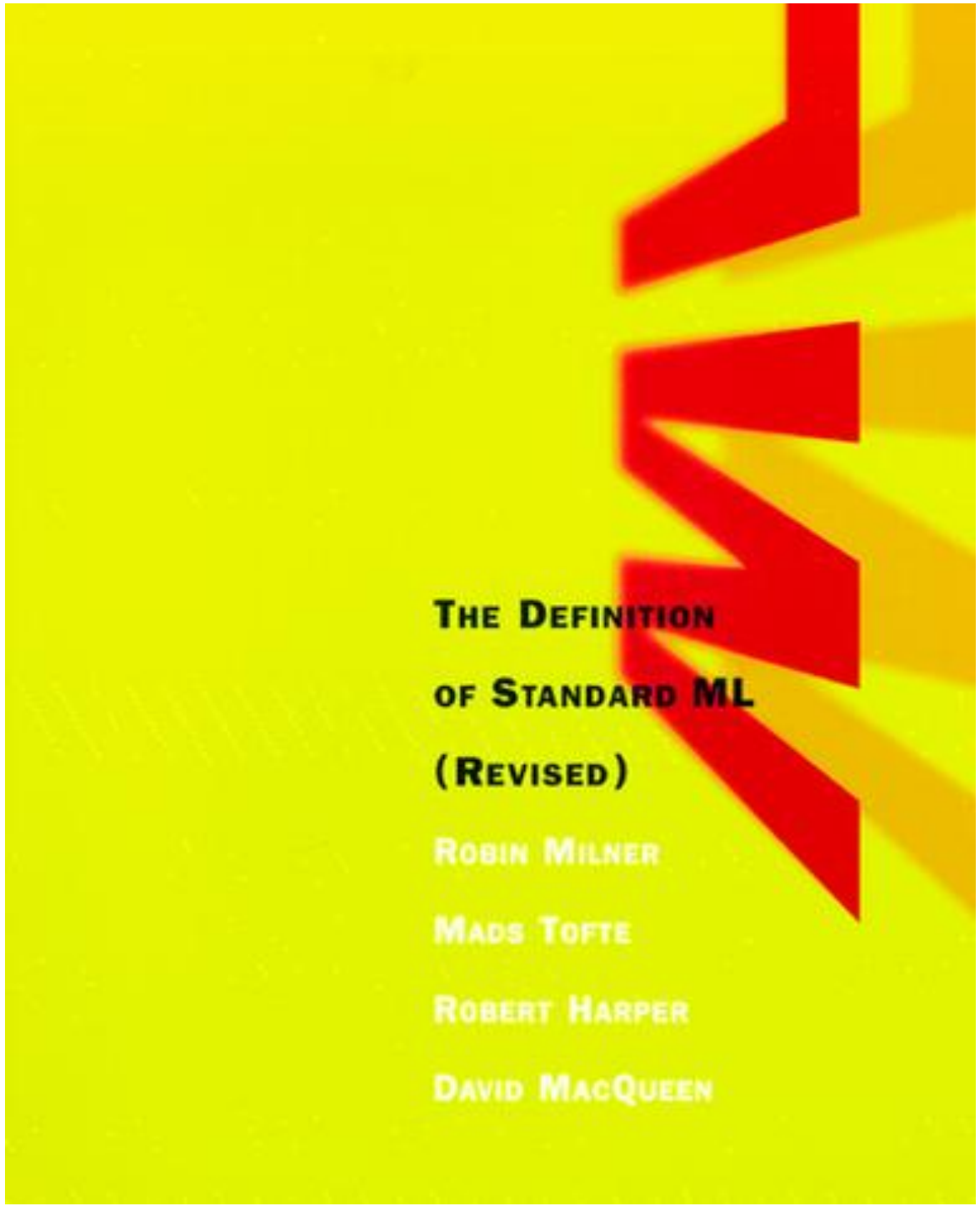
Latest news: Poly/ML Version 5.4.1 has now been released.

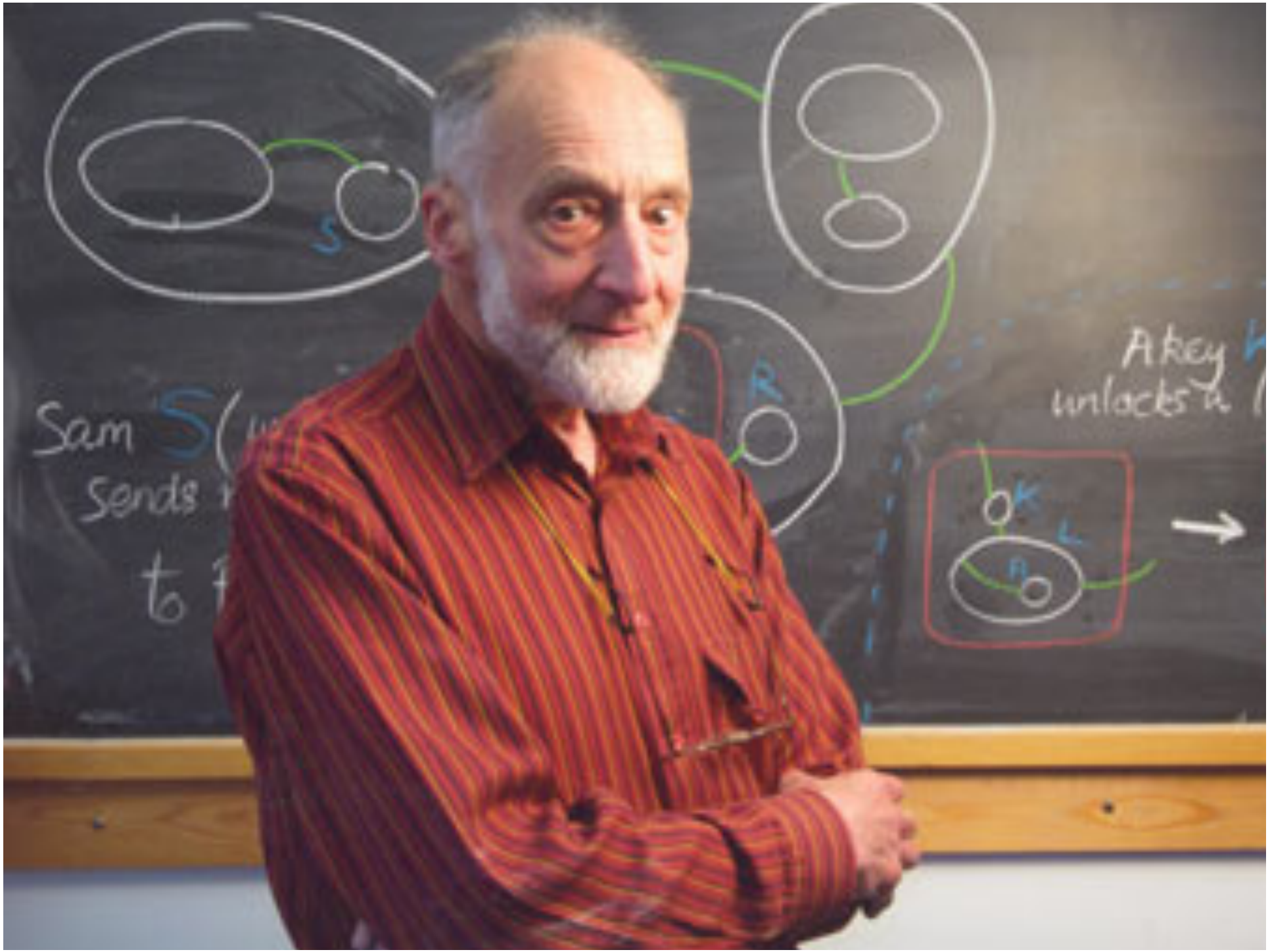
There is a Poly/ML mailing list available at the University of Edinburgh.

Last

polyml.org


sourceforge
Find Open Source Software
Browse
Blog
Support
Jobs
Home / Browse / Poly/ML
Summary
Files
Reviews
Support
Develop
Hosted Apps
Tracker
Code
Poly/ML
dcjm
4 Recommendations
19 Downloads (This Week)
Download
polyml.5.4.1.tar.gz
Tweet
0
+1
0
Like
Description
sourceforge.net/projects/polyml/


THE DEFINITION
OF STANDARD ML
(REVISED)
ROBIN MILNER
MADS TOFTE
ROBERT HARPER
DAVID MACQUEEN


S
R
Sam
Sends
to
A key
unlocks a
K
L
R

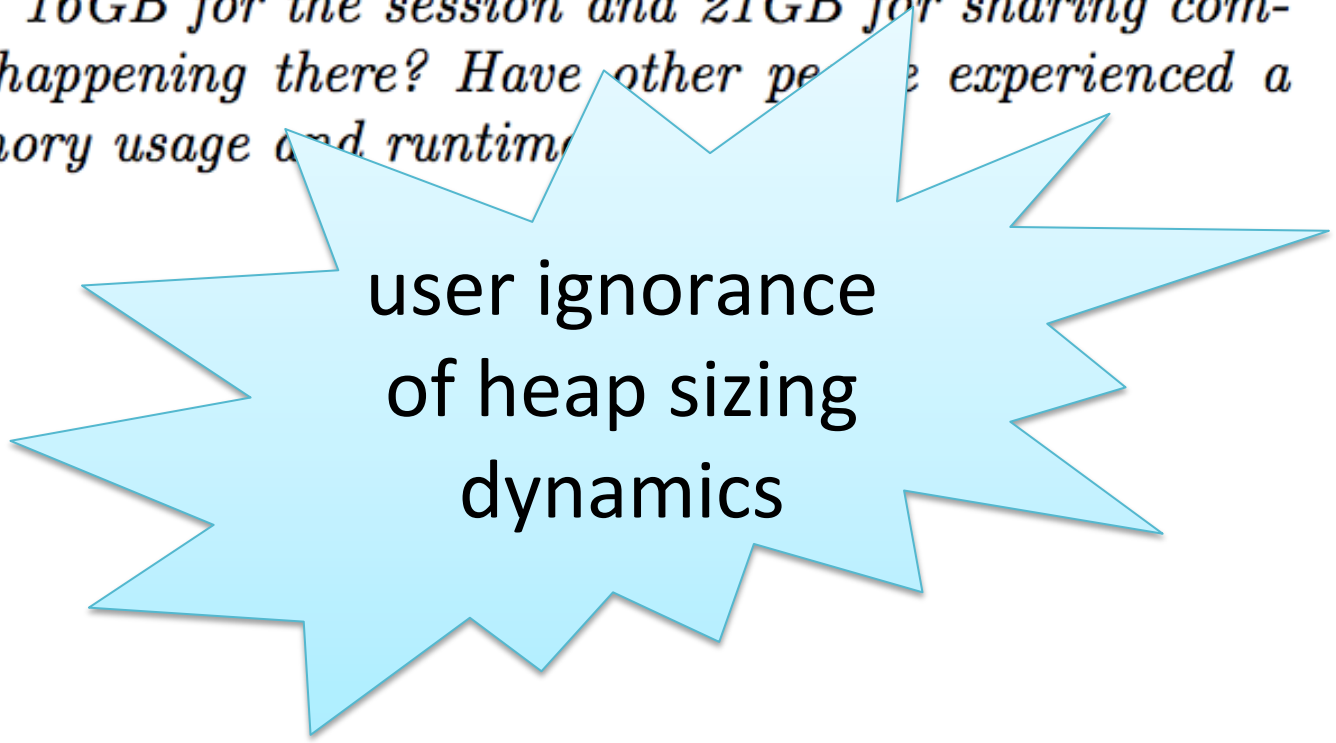

Objective Caml


Microsoft
Visual F#


Isabelle
∀
=
⊢
α
λ
β
→

When I build a large session in Isabelle2011-1 (to be precise, an extension of of JinjaThreads in the Archive of Formal Proofs) with PolyML 5.4.1, this takes 1:51h. While running the session, polyml requires 12GB of memory (`VmSize` in `/proc/PID/status`). The final call to `PolyML.shareCommonData` before writing the heap image consumes 17GB of memory.

I now ran the same session with the SVN version 1352 of PolyML. Then, it takes 2:35h, 16GB for the session and 21GB for sharing common data. What is happening there? Have other people experienced a similar surge in memory usage and runtime?



What are the RTS options?

- ./poly --help

Run time system arguments:

-H <Initial heap size (MB)>

--heap <Initial heap size (MB)>

--immutable <Initial size of immutable buffer (MB)>

--mutable <Initial size of mutable buffer(MB)>

--allocation <Size of allocation area(MB)>

--stackspace <Space to reserve for thread stacks and C++ heap (MB)>

--gcthreads <Number of threads to use for garbage collection>

--debug <Debug options: checkmem, gc, x>

--logfile <Logging file (default is to log to stdout)>

--gcshare <Perform a data sharing pass before each major GC>

--resizing <Policy for RTS heap resizing: default, fixed>

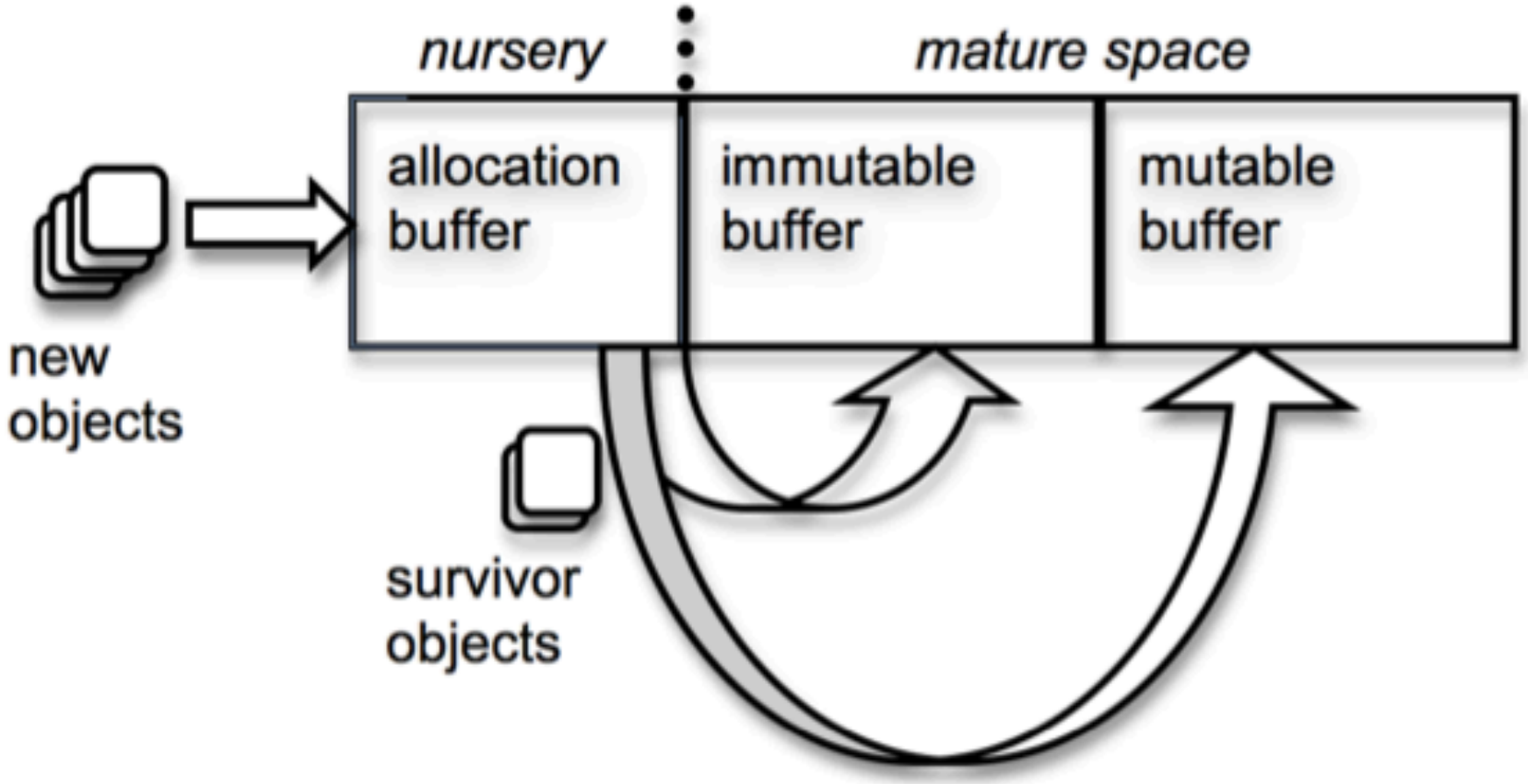


Fig. 2. Schematic Layout of the Poly/ML Runtime Heap

What is Poly/ML's default heap resizing policy?

After major GC, call **`adjustHeapSize()`**

```
matureBudget =
    liveData + SPARE_SPACE
```

`nurseryBudget` unchanged

Why is heap resizing important? (1)

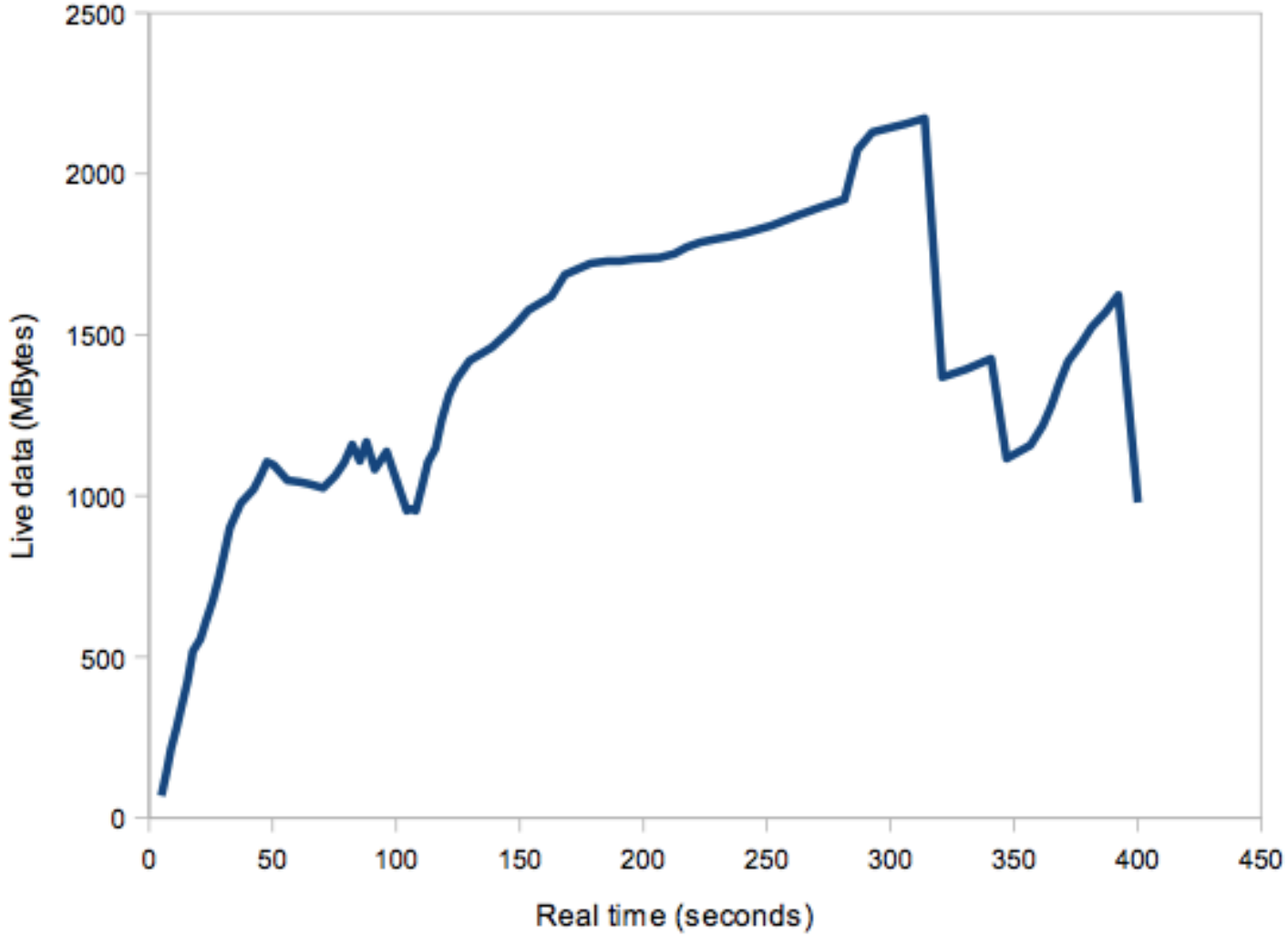
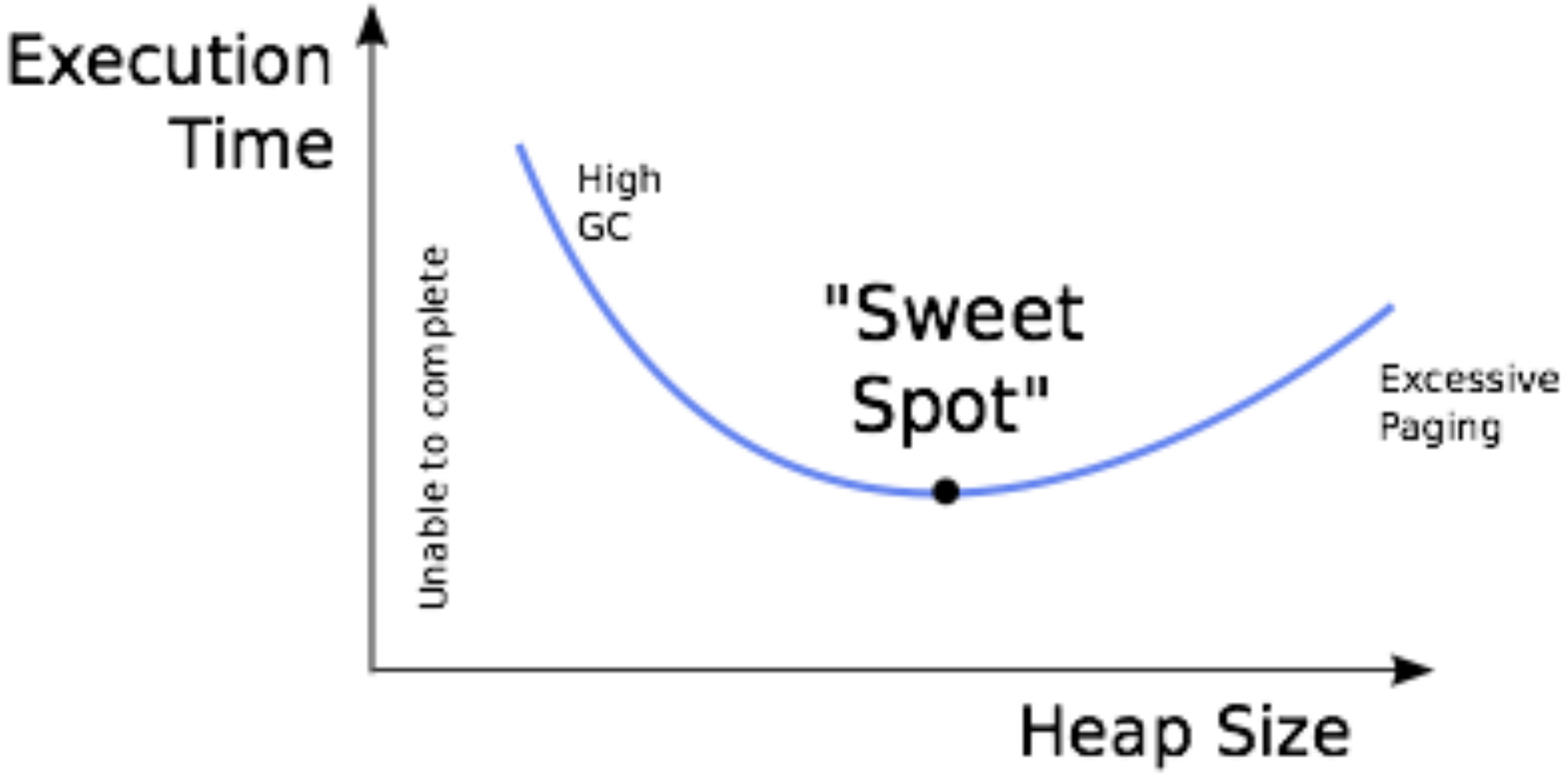


Fig. 3. Graph showing total live data size measured at each major GC against real-time for a single execution of a typical Poly/ML workload

Why is heap resizing important? (2)



Criticism of Poly/ML's default heapsizing policy

- No fixed bound on memory consumption
- Lack of sophistication in growth

Our plans

- add a **`--resizing`** option to command line
- selection of policies:
  - **`default`**
  - **`fixed`**
  - **`pid`**

Fixed (Bounded) Heap Sizing

- At every point in RTS where new space can be allocated, do an explicit check to see whether new space would be within budget

- invariant:

```
nurseryBudget + matureBudget ==
      FIXED_HEAP_BUDGET
```

- Distinction between budget and current usage

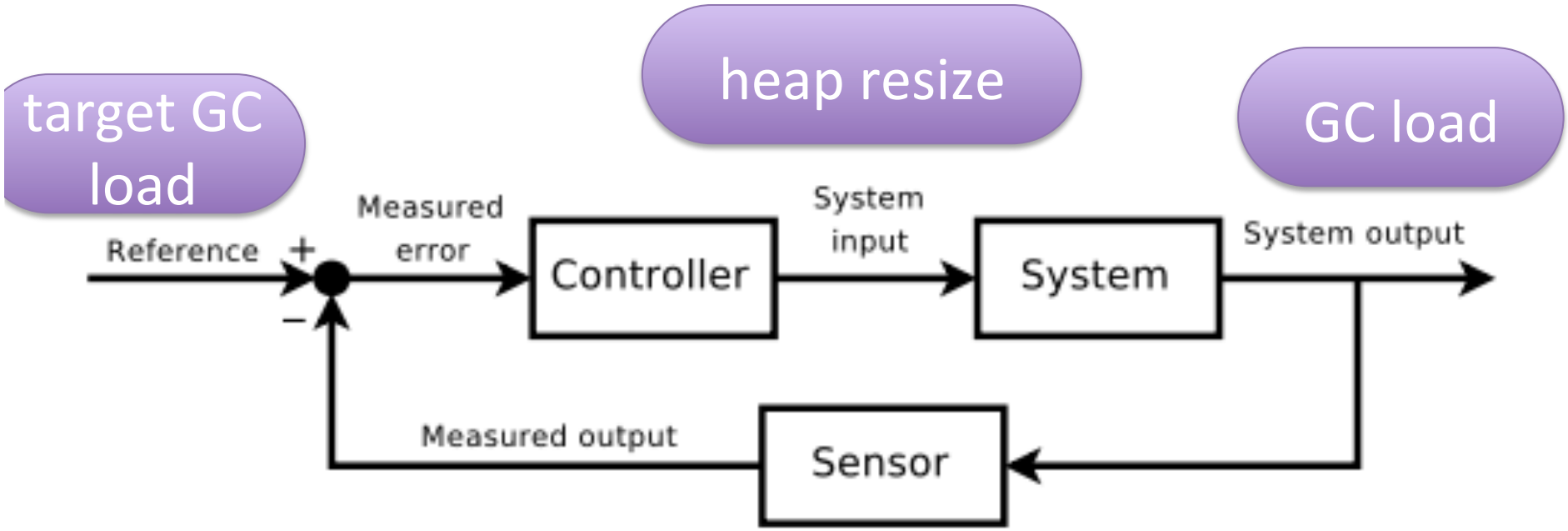
Status: experimental, in progress

- http://polyml.svn.sourceforge.net/viewvc/polyml/branches/heapsizing-branch/

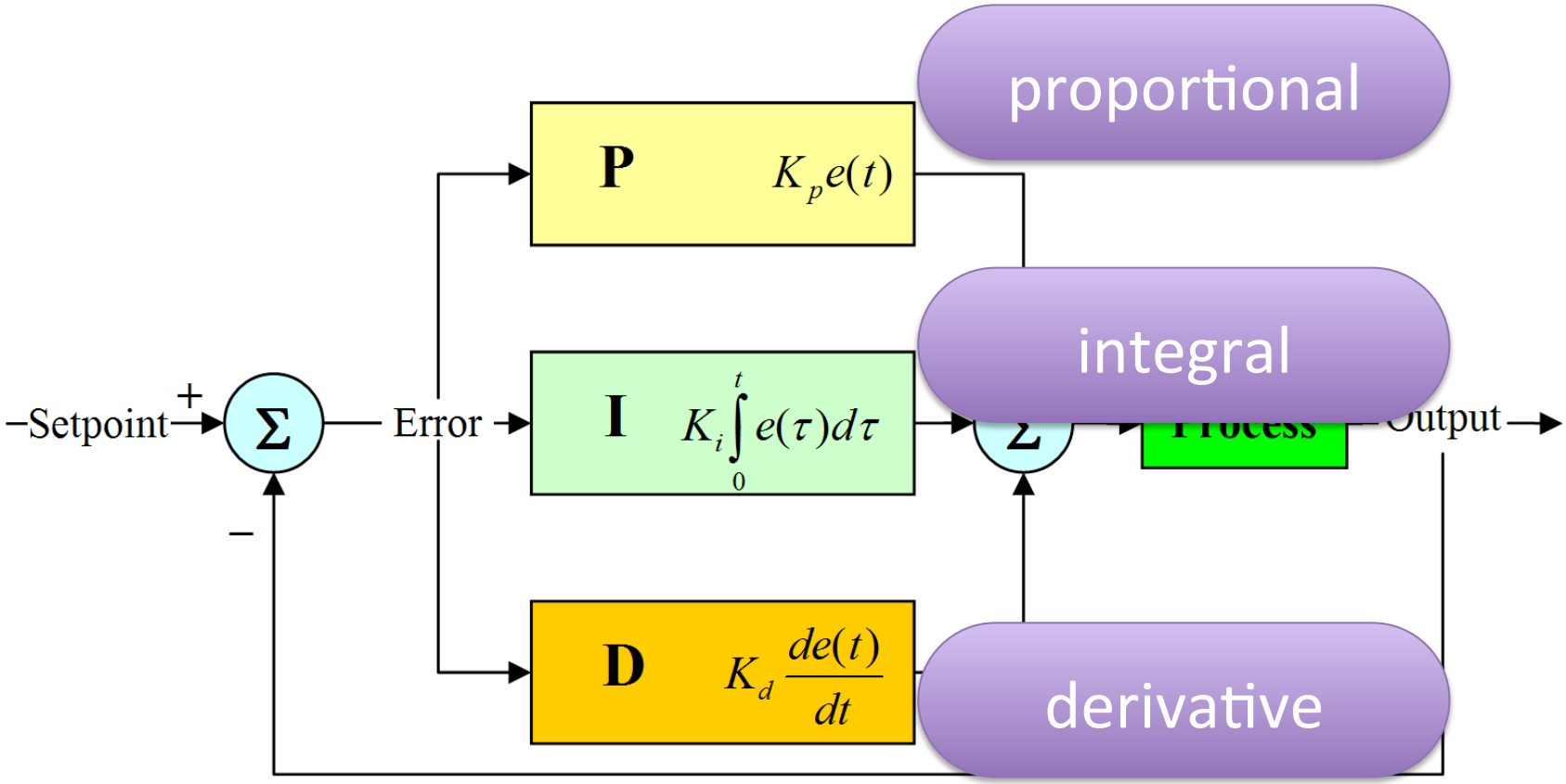
PID controller for heap sizing

- Control theory 101

Control theory *a la* Wikipedia



PID controller



PID resizing implementation - status

- Have a PID controller coded up
- Hooked up with **`adjustHeapSize()`**
- Needs parameter *tuning*
- Needs performance validation
- Needs testing in the wild
  - (on Isabelle theorem proving benchmarks?)

How do other VMs handle heap growth?

GHC runtime system

heap size multiplicative increase if ‘nearly full’

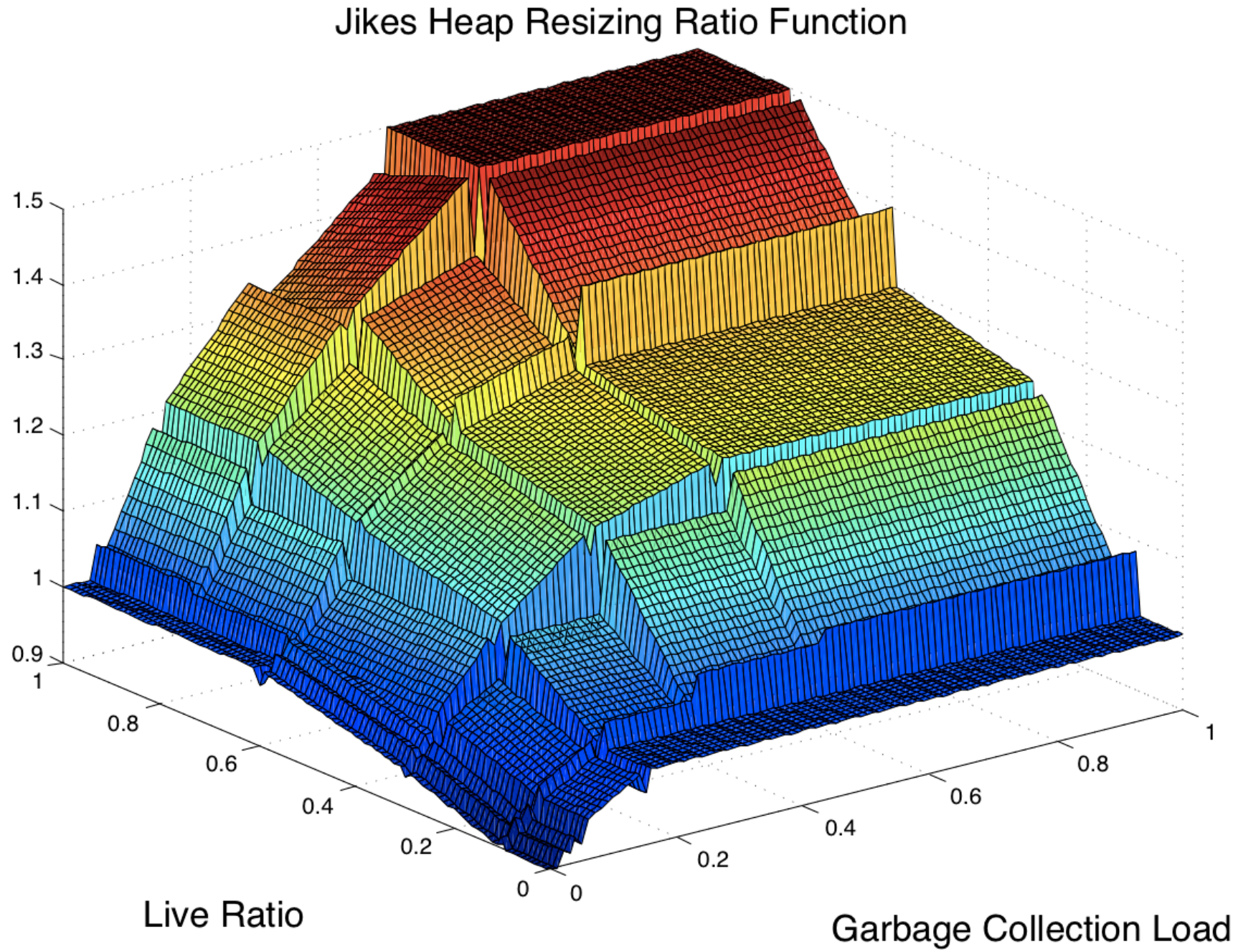
[Simon Marlow]

Jikes RVM

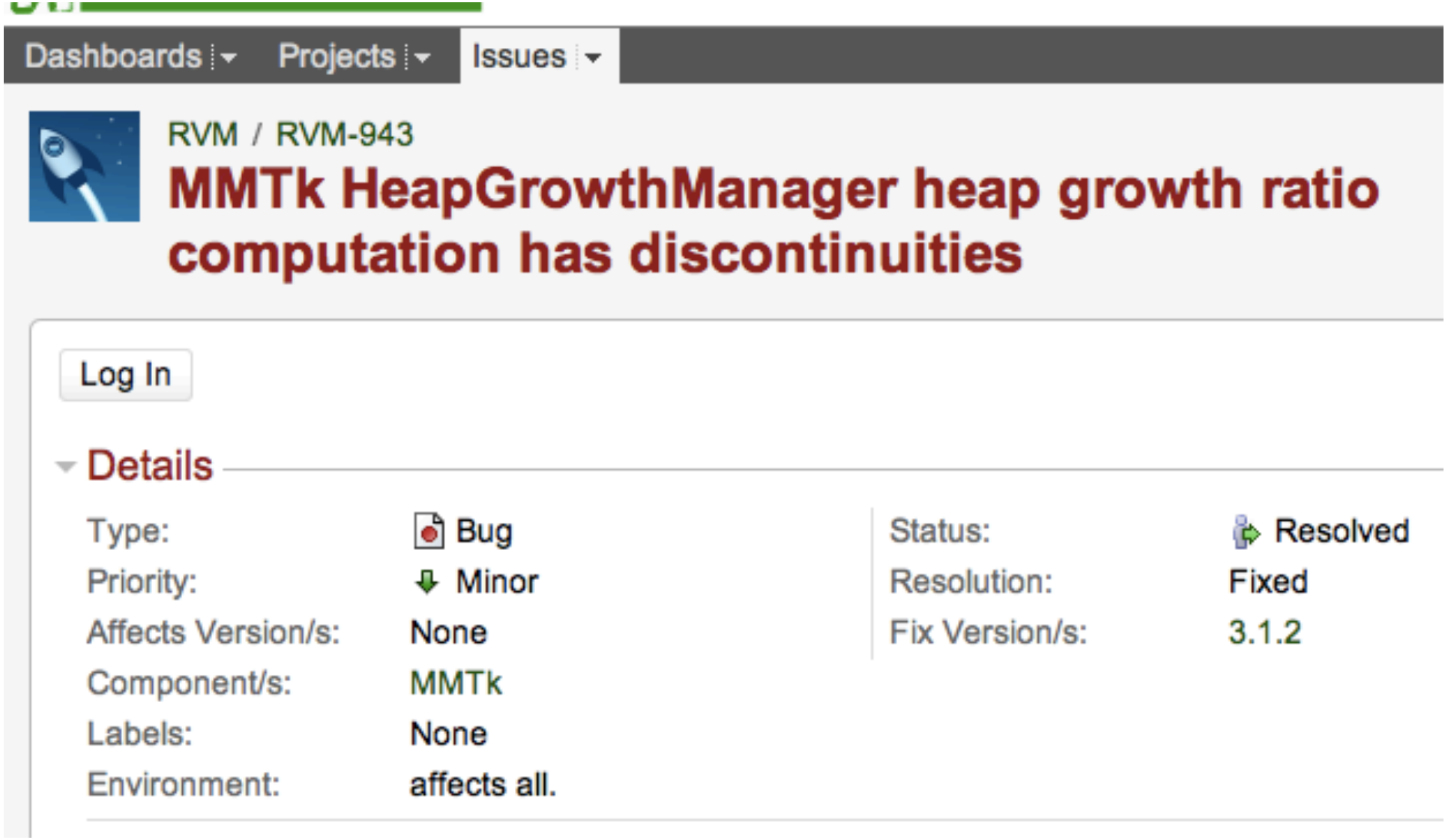
- **`HeapGrowthManager`** computes heap resize ratio after major GC
- 2d lookup table of resize ratios, indexed by:

$$g = \frac{\text{Time taken for most recent collection}}{\text{Time since last collection}}$$

$$l = \frac{\text{Amount of live data on the heap}}{\text{Current heap size}}$$


Jikes Heap Resizing Ratio Function
1.5
1.4
1.3
1.2
1.1
1
0.9
1
0.8
0.6
0.4
0.2
0
Live Ratio
0
0.2
0.4
0.6
0.8
1
Garbage Collection Load

Bug fix


Dashboards
Projects
Issues
RVM / RVM-943
MMTk HeapGrowthManager heap growth ratio computation has discontinuities
Log In
Details
Type:
Bug
Priority:
Minor
Affects Version/s:
None
Component/s:
MMTk
Labels:
None
Environment:
affects all.
Status:
Resolved
Resolution:
Fixed
Fix Version/s:
3.1.2

- In HeapGrowthManager.computeHeapChangeRatio(), the current implementation determines the heap size change ratio by a lookup in the 2-dimensional function table (indexed by liveRatio and gcLoad). Given a current liveRatio X and gcLoad Y, the code finds the table rows and columns with nearest values above and below X and Y, then does interpolation from these table lookup values to determine the heap size change ratio.
- However, there is a bug in the interpolation. If X (or Y, respectively) is exactly equal to a row (or column, respectively) label value, then the interpolation still happens, between values in rows (cols) either side of row X (col Y). This leads to discontinuities in the heap sizing function - see attached graphs.
- The submitted patch suppresses interpolation (interpolation correction value becomes 0) in the case where X or Y falls on a label value exactly, so avoiding the discontinuity.

OpenJDK

- GC ergonomics system allows user to specify high-level goals for GC
  - desired max GC pause time
  - desired application throughput
  - minimum heap size

OpenJDK

- **`AdaptiveSizePolicy`** applies fixed rules to give best-effort satisfaction of these targets:
  - if current pause time > pause time goal, then *decrease* heap size
  - else if application's throughput goal is not being met, then *increase* heap size
  - else *decrease* heap size to reduce memory footprint

OpenJDK

- David Vengerov [ISMM11]
- [The ergonomics system consists of] some heuristic rules that do not guarantee that the GC throughput will actually be maximised as a result

Dalvik

- Easy to grow heap, more difficult to shrink
  - non-moving objects in mature space

$$h' = \frac{\text{current size of live data}}{\text{target utilization rate}}$$

- Heap sizing policy entirely opaque to user

Dalvik heap resizing app

Find the best Android apps | Hot today | Hot this week | All-time popular | Top rated

All apps » Tools » VM Heap Tool (root only)

VM Heap Tool (root only)

by martino

Install Free

50,000-250,000 downloads, **365** ratings (4.50 average), 51 kb, Permissions, Official Page, Contact

Add to list ▼ Like +1 0 Tweet 2 Email QR / more ▼

Requires *FULL ROOT* (aka S-OFF or NAND unlock) and *BUSYBOX*. Please use Busybox version 1.17.x as 1.18.x compile is broken!

Allows to change the Dalvik VM heap

Heuristics have problems

replace heuristics with

mathematical models

How to show that our scheme is better?

- What does *better* mean?
  - performance in terms of execution time?
  - resource use in terms of memory space rental?

Final word

- ongoing implementation effort
- looking for users/testers
- feedback welcome
- extending ideas to other systems
- Big Picture:
  - replacing heuristics with mathematical models

University of Glasgow

Please get in touch and help out …

http://www.dcs.gla.ac.uk/~jsinger

Jeremy.Singer@glasgow.ac.uk

@jsinger_compsci