Java Heap Resizing

From Hacked-up Heuristics to Mathematical Models

Jeremy Singer and David R. White

University of Glasgow | School of Computing Science

Outline

Outline

Cloud

Datacentres

Key Question

To be *economical*, can we *over-subscribe* resources?

Key Question

To be *economical*, can we *over-subscribe* resources?

i.e. particularly dynamic memory consumption

Current Jikes Policy: Resize Matrix

		Heap Occupancy					
		0.00	0.10	0.30	0.60	0.80	1.00
GC Overhead	0.00	0.90	0.90	0.95	1.00	1.00	1.00
	0.01	0.90	0.90	0.95	1.00	1.00	1.00
	0.02	0.95	0.95	1.00	1.00	1.00	1.00
	0.07	1.00	1.00	1.10	1.15	1.20	1.20
	0.15	1.00	1.00	1.20	1.25	1.35	1.30
	0.40	1.00	1.00	1.25	1.30	1.50	1.50
	1.00	1.00	1.00	1.25	1.30	1.50	1.50

Look-up table for heap-resize coefficient.

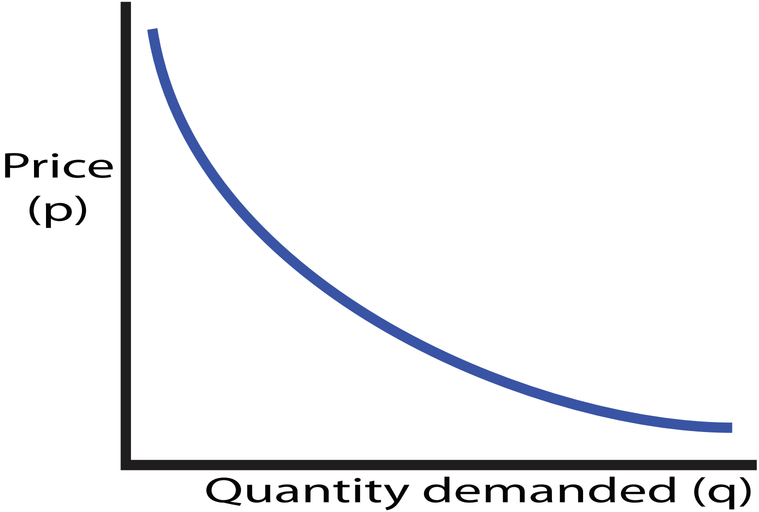
Design Decisions

(Private Communication)

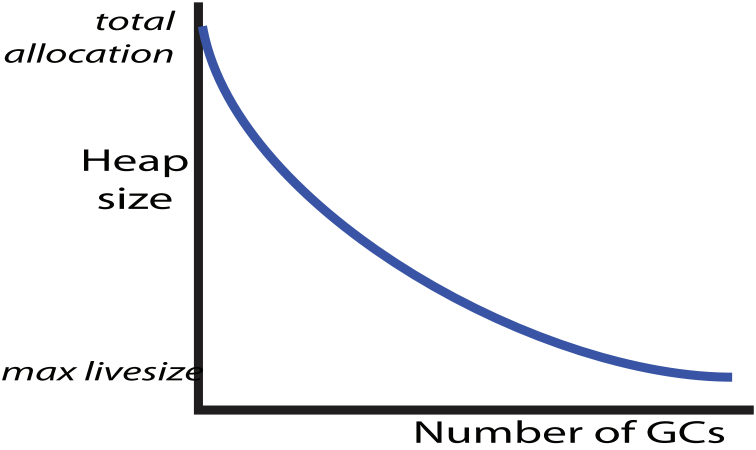
> *"...back in 2003 [anon] and I did some experimental tuning and came up with the numbers by eyeballing things. At the time, it seemed to be somewhat stable and making reasonable decisions but that was also about 4 major versions ago and I don't think anyone has really looked at it seriously since then. I think there was some amount of sensitivity to the values..."*

Outline

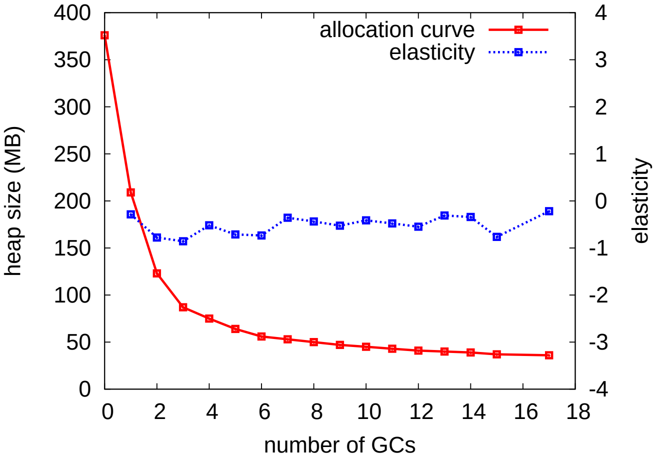
Demand Curve



Allocation Curve



Empirical Data



Demand Elasticity

- E measures sensitivity of the quantity q demanded to changes in price p.

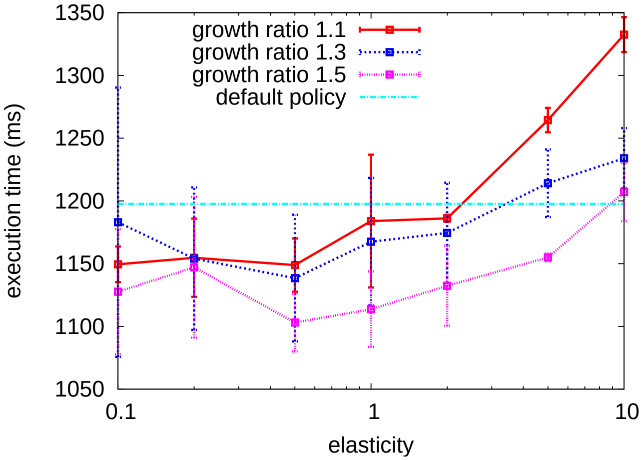
$$E = \frac{\text{\% change in quantity}}{\text{\% change in price}} = \frac{dq}{dp}\frac{p}{q} \tag{1}$$

Allocation Elasticity

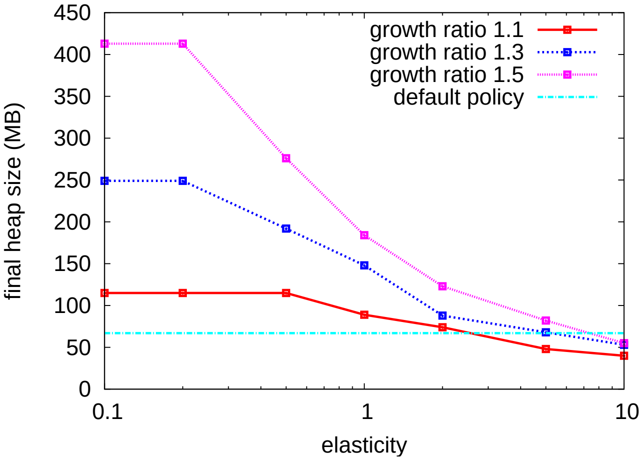
- E measures sensitivity of number of GCs g demanded to changes in heap size h.

$$E = \frac{\text{\% change in num GCs}}{\text{\% change in heap size}} = \frac{dg}{dh}\frac{h}{g} \tag{2}$$

Control heap size with elasticity parameter



Control heap size with elasticity parameter



Problems with elasticity-based heap growth

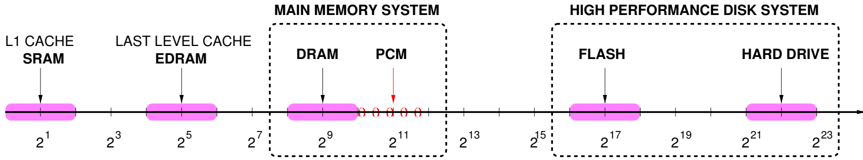
- unintuitive parameter for users
- difficult to bound heap growth

Problems with elasticity-based heap growth

- unintuitive parameter for users
- difficult to bound heap growth
- . . . problems with paging!

Relative costs of memory accesses

We must avoid paging at all costs!

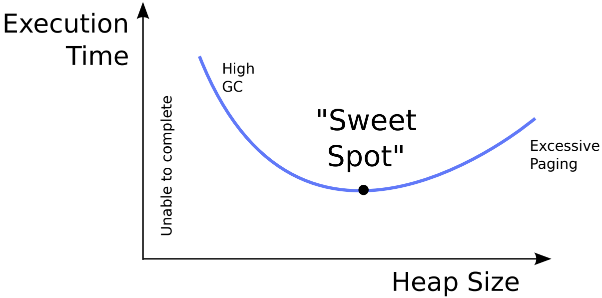


Typical Access Latency (in terms of processor cycles for a 4 GHz processor)

Scalable High Performance Main Memory System Using Phase-Change Memory Technology.
Qureshi et al.

Finding the Sweet Spot

If Goldilocks did heap sizes. . .



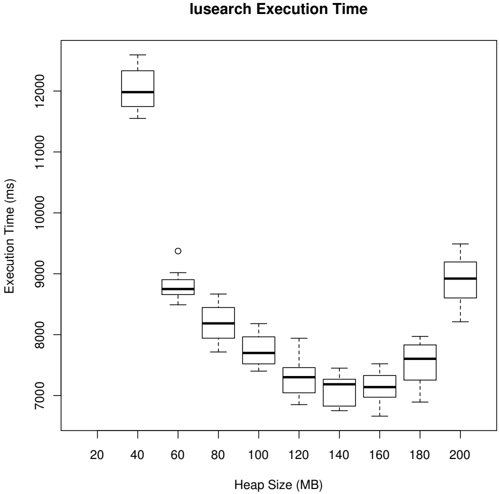
Experiment

- Linux
- Single-user mode.
- 300MB System RAM (via Kernel Parameter)

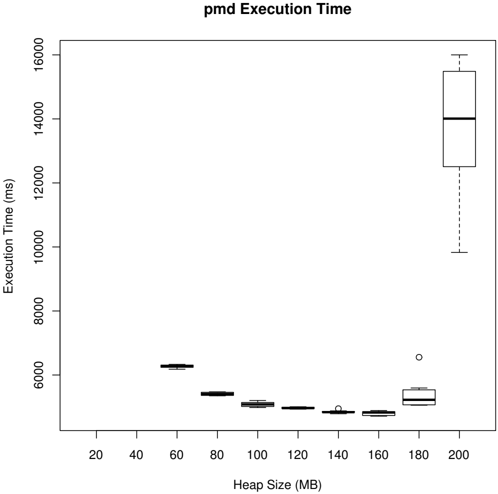
Used the Jikes RVM and the Da Capo Benchmarks.

Plot Heap Size versus Execution Time.

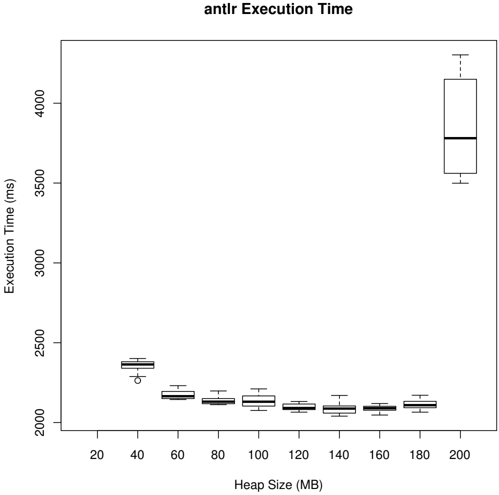
Results: Heap Size versus Execution Time



Results 2: Heap Size versus Execution Time



Results 3: Heap Size versus Execution Time



Key Question

How large should an application heap be?

Proposal?

Yet

Another

Heuristic

Why Not? Strong Foundations

Although the problem can be regarded as one of 'best effort', there are some properties we want our system to have:

- Guarantees about convergence.
- No pathological behaviours.

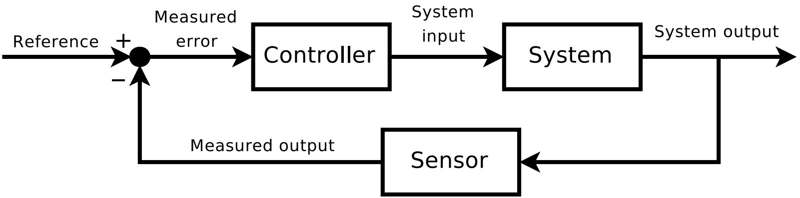
Therefore: tractability is important.

Problem Summary

Create a device that constantly pushes the virtual machine towards the optimal heap size, in the presence of variation in software behaviour and disturbances from external factors such as other applications and the operating system.

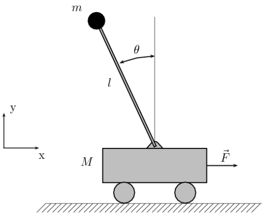
Outline

A Control System

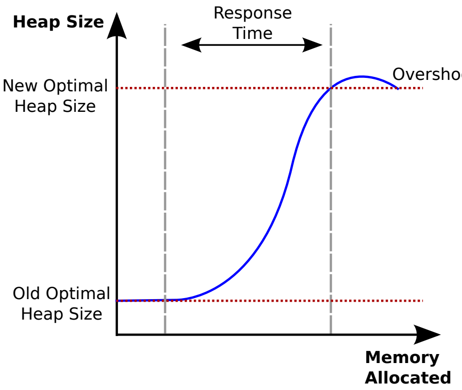


Example Control Systems

Example Control Systems



Viewing Heap Sizing as a Control Problem



Issues with Applying Control Theory to Heap Sizing

1. Time is variable.

Issues with Applying Control Theory to Heap Sizing

1. Time is variable.
   $\rightarrow$ Use memory allocated as a proxy.

Issues with Applying Control Theory to Heap Sizing

1. Time is variable.
   $\rightarrow$ Use memory allocated as a proxy.
2. Control and Process Variables.

Issues with Applying Control Theory to Heap Sizing

1. Time is variable.
   $\rightarrow$ Use memory allocated as a proxy.
2. Control and Process Variables.
   $\rightarrow$ Control variable is heap size (absolute, relative/ratio?)
   $\rightarrow$ Process: GC Overhead (long/short-term), Occupancy, Mark-Cons ratio ...

Issues with Applying Control Theory to Heap Sizing

1. Time is variable.
   $\rightarrow$ Use memory allocated as a proxy.
2. Control and Process Variables.
   $\rightarrow$ Control variable is heap size (absolute, relative/ratio?)
   $\rightarrow$ Process: GC Overhead (long/short-term), Occupancy, Mark-Cons ratio . . .
3. Specific to our system
   e.g. exactly when do we allow heap-resizing to happen.

Issues with Applying Control Theory to Heap Sizing

1. Time is variable.
   $\rightarrow$ Use memory allocated as a proxy.
2. Control and Process Variables.
   $\rightarrow$ Control variable is heap size (absolute, relative/ratio?)
   $\rightarrow$ Process: GC Overhead (long/short-term), Occupancy, Mark-Cons ratio ...
3. Specific to our system
   e.g. exactly when do we allow heap-resizing to happen.
4. Characterising our system.

Issues with Applying Control Theory to Heap Sizing

1. Time is variable.
   $\rightarrow$ Use memory allocated as a proxy.
2. Control and Process Variables.
   $\rightarrow$ Control variable is heap size (absolute, relative/ratio?)
   $\rightarrow$ Process: GC Overhead (long/short-term), Occupancy, Mark-Cons ratio . . .
3. Specific to our system
   e.g. exactly when do we allow heap-resizing to happen.
4. Characterising our system.
   $\rightarrow$ Using existing models.
   $\rightarrow$ Applying empirical analysis.

Using Existing Analytical Work

$$n = \begin{cases} n^* & \text{for } M \geq M^* \\ \frac{1}{2}(K + \sqrt{K^2 - 4})(n^* + n_0) - n_0 & \text{for } M < M^* \end{cases}$$

where $K = 1 + \frac{M^* + M^0}{M + M^0}$

n^*, M^*, M^0, n_0 are parameters dependent on the software and platform.

A Page Fault Equation for Dynamic Heap Sizing. Tay and Zong.

Implementation so far: PID Controller

Control signal $u(t)$ governed by the following equation:

$$u(t) = K_P e(t) + K_i \int e(t)dt + K_D \frac{d}{dt} e(t) \tag{3}$$

Model Tuning

For example, how do we decide upon the coefficients K_P, K_i, K_D?

Still an optimisation problem: but built upon a well-formulated control problem.

Outline

Quick Summary

- Heap sizing is an important factor in determining performance.
- Determining optimal heap size is a difficult, dynamic problem.
- Control theory gives us a way to approach it systematically.

Future Work

Revisiting analysis and determining the correct controller.

Optimisation schemes.

Empirical Evaluation.