APPLIED COMPUTING, MATHEMATICS AND STATISTICS GROUP
Division of Applied Management and Computing

Displaying Cache Information in the JADE Object Oriented Distributed Processing System

A.E. McKinnon and R. Jarquin

Research Report No: 99/06
March 1999

ISSN 1174-6696

RESEARCH REPORT

LINCOLN UNIVERSITY
Te Whare Wānaka O Aoraki

Applied Computing, Mathematics and Statistics

The Applied Computing, Mathematics and Statistics Group (ACMS) comprises staff of the Applied Management and Computing Division at Lincoln University whose research and teaching interests are in computing and quantitative disciplines. Previously this group was the academic section of the Centre for Computing and Biometrics at Lincoln University.

The group teaches subjects leading to a Bachelor of Applied Computing degree and a computing major in the Bachelor of Commerce and Management. In addition, it contributes computing, statistics and mathematics subjects to a wide range of other Lincoln University degrees. In particular students can take a computing and mathematics major in the BSc.

The ACMS group is strongly involved in postgraduate teaching leading to honours, masters and PhD degrees. Research interests are in modelling and simulation, applied statistics, end user computing, computer assisted learning, aspects of computer networking, geometric modelling and visualisation.

Research Reports

Every paper appearing in this series has undergone editorial review within the ACMS group. The editorial panel is selected by an editor who is appointed by the Chair of the Applied Management and Computing Division Research Committee.

The views expressed in this paper are not necessarily the same as those held by members of the editorial panel. The accuracy of the information presented in this paper is the sole responsibility of the authors.

This series is a continuation of the series "Centre for Computing and Biometrics Research Report" ISSN 1173-8405.

Copyright

Correspondence

This paper represents work to date and may not necessarily form the basis for the authors' final conclusions relating to this topic. It is likely, however, that the paper will appear in some form in a journal or in conference proceedings in the near future. The authors would be pleased to receive correspondence in connection with any of the issues raised in this paper. Please contact the authors either by email or by writing to the address below.

Any correspondence concerning the series should be sent to:

The Editor
Applied Computing, Mathematics and Statistics Group
Applied Management and Computing Division
PO Box 84
Lincoln University
Canterbury
NEW ZEALAND

Email: computing@lincoln.ac.nz

Displaying Cache Information in the JADE Object Oriented Distributed Processing System

A.E. McKinnon,
Applied Management and Computing Division
Lincoln University
Email: mckinnon@lincoln.ac.nz

R. Jarquin,
Aoraki Corporation
Christchurch, New Zealand
Email: rjarquin@cnzmail.aoraki.co.nz

Introduction

One of the major factors affecting the performance of any distributed processing system is the management of the cache in the local workstation. JADE is an object oriented distributed processing system developed by Aoraki Corporation of Christchurch, New Zealand. JADE uses a fully object model both for its underlying database and its development environment. Much of JADE is itself written in the JADE language.

A distributed transaction processing system such as JADE must on the one hand provide for full data integrity using appropriate locking and database management techniques and on the other hand ensure that local cache is managed in such a way that users experience the best performance possible and the network is not unduly loaded. This is made more complex because the flexibility of JADE means that application code (methods) can run on either the client or the server depending on where the developer perceives it will run more efficiently.

The basic unit of caching in JADE is an individual object. Each object occupies a *buffer* in the cache memory. At present JADE uses a "least recently used" (LRU) cache-ordering algorithm (Stallings, 1993, p. 163) in which the last cache buffer referenced is placed at the top of the cache. When space is required for a new buffer those at the bottom are displaced first.

This paper is an initial attempt at providing a visual description of the arrangement of objects in the cache, based on *snapshots* of cache information taken while a JADE application is run.

Access to Cache Data

Cache management is not a new problem and neither is the provision of visualisation tools to help assess cache behaviour (van der Deijl *et al*, 1997). The problem common to all cache environments is the large amount of rapidly changing data involved. Although some systems provide a view of the cache under study in real time (van der Deijl *et al*, 1997) we have implemented a scheme where data snapshots are gathered while the system is running but the analysis is done off-line. This eases implementation within JADE and ensures that the cache viewing has minimal impact on the performance of the running system.

To achieve this R.Jarquin modified the JADE kernel to include some code which, when signalled by the server using JADE's notification system, causes JADE to write a snapshot of its local cache to a disk file. This process is very quick and has imperceptible impact on the performance of the client node. Such snapshots can be taken as frequently as desired and then loaded into a separate JADE database for further analysis.

An overview of the information recorded in each snapshot file is given in Appendix A. In each client there is a transient cache and a persistent cache. The former is for temporary objects, which are created by the client application and the latter for information, which either came from or must be sent to the server. The information stored in either cache is stored in buffers, which are created dynamically within the cache memory. Each buffer is classified according to its physical type. For example an object might be a database object, and its semantic type might be a GUI (Graphical User Interface) object.

Viewing the Cache Snapshots

The prototype system developed for viewing the cache snapshots assumes that the transient and persistent caches will be viewed separately. In its prototype form, the system shows one snapshot at a time in the following manner (refer to Figure 1):

- each buffer is represented by a square group of pixels on the main part of the display .
- buffers are grouped according to their physical type or their semantic type as specified by the user. For example in Fig 1a all the *Normal* buffers are grouped together to the right of the red vertical line at the left of the display. All the *Collection Header* buffers are grouped together to the right of the dark blue vertical line, all the *Collection Block* buffers are grouped together to the right of the yellow vertical line and the *Blob/slob* buffers are grouped together to the right of the magenta vertical line. This means that the area of the display between the strongly coloured vertical lines represents the **number** of buffers of the corresponding type.
- the **order** of each buffer within the cache, in the LRU sense, is indicated by the colour of the pixels used to display it. Buffers are shown in order from most recently used to least recently used within each type, going from top to bottom and left to right of the display screen. The most recently used buffers are shown in the darkest colour. Note that this colouring applies independent of buffer type. For example, in Figure 1a the colouring indicates that there are buffers of all types which have been recently used (dark colour) but the absence of very light colour for *Blob/slob* buffers indicates that buffers of this type have not been unused in the cache for as long as buffers of other types.
- above the display of buffers is a simple bar chart. The height of each bar is proportional to the sum of the **sizes** of all the buffers in the column of buffers which is directly underneath that bar. This is normalised across the whole display. The value corresponding to the height of the tallest bar is shown beside the *Size of column of buffers. Max=* heading. For Fig 1 a the value is 185.56 kbytes.

A Test Snapshot

Figure 1 shows the displays from a test snapshot, which was taken from an internal company benchmark called TPC during the application loading phase. The figure shows the display of physical and semantic buffer types for both the persistent and transient caches as described above. The identifier for each buffer type is shown below the buffer display in a colour matching that of the corresponding vertical line. The displays shown in the figure are a compressed version of those actually displayed on the screen.

Interpreting the Displays

The displays shown in Figure 1 show some information about the objects in the cache at the time the snapshot was taken. They clearly show the relative numbers of buffers of each type and also, because of the colour distribution, their relative positions within the cache are evident.

The actual snapshot displayed in Figure 1 was obtained primarily for testing purposes and is no way representative of what might happen in general. JADE has both *persistent objects* which are stored in the database as the application's persistent or permanent data and *transient objects* which may be created as the application runs but are not stored permanently. There are separate caches for persistent and transient objects. Each object has both a *physical* type which describes the nature of the physical information it

contains (such as a normal object or a binary large object – Blob) and a *semantic* type which describes its role or origin (such as a user object, a system meta object or a user GUI object).

The persistent cache buffers (Figure 1a) are mostly of *Normal* type although each type has buffers fairly well distributed throughout the cache in the sense of LRU ordering. Perhaps not surprisingly, the size bar chart shows that the bigger buffers are *Collection Blocks* and *Blob/slobs*. The larger buffers are found in one column of type *Blob/slob*. The semantic type of the persistent buffers is predominantly *System Meta* (Figure 1b) and the group of buffers taking the most memory are of type *System GUI*. According to the colour, this group is about half way down the cache in an LRU sense. There are essentially no user objects of any type in this snapshot.

There are fewer transient buffers than persistent (Figures 1c and 1d are smaller than Figures 1a and 1b). The physical types are distributed throughout the cache with the largest amount of space being taken by a *Blob/slob*. The *Unknown* type refer to buffers that have been deleted but whose space has not yet been recovered by garbage collection. There are virtually no user buffer types in the transient cache.

The test snapshot shown in Figure 1 is relatively small in terms of the number of buffers. Typically, a cache in JADE will contain 3,500 buffers per MB so that a large cache might contain up to 30,000 buffers. Given the display arrangement shown in Figure 1, this would mean a maximum dot of 3x3 pixels to display one buffer if all buffers are to fit on the screen at the same time. Dots this small could be very hard to distinguish on some display screens.

Related to the above is the size and appropriateness of the colour maps involved. The colour palette provided by a computer's graphics card may be limited to 256 colours. With this lack of colour resolution, even if there is sufficient physical space on the display to allow all buffers to be displayed as, say, squares of 3x3 pixels, adjacent pixels will not have different colours. That is evident in Figure 1. At this stage, it is not clear how important it is to be able to see individual buffers in the displayed image. Experience with a range of snapshots for a variety of JADE applications will help to determine this.

An enhancement that would improve the readability of each display would be to replace the LRU colour legend in the lower part of the image with a series of tick marks placed under the buffer display. The tick marks would be positioned under each buffer type at intervals corresponding to 10% of the total number of objects in the cache. The tick marks would essentially be marking positions of common colour between buffer types.

The information shown in figure 1 is by no means all that is available about the objects stored in the buffers in JADE's cache. In particular circumstances it might be desirable to be able to access all the details that are listed in Appendix A about an individual buffer. This could be readily achieved by pointing to the buffer of interest on the display and having the details displayed. This assumes, of course, that the display is such that buffers can be distinguished.

Thus far the discussion has related to only one snapshot. When JADE's buffer management is being investigated for a particular application, it is likely that a sequence of snapshots would be taken. Although each snapshots would be able to be viewed as in Figure 1, further information about cache management might be obtained by considering where particular objects appeared in the cache throughout the sequence of snapshots. Of particular interest would be objects which were present in a snapshot, absent from the next one in the sequence and then present in a later one.

Using overlapping windows, the display technique described here could be used for multiple snapshots. It would be possible for the movement of a selected object to be traced through the series of snapshots by appropriate highlighting. The program could automatically search for and highlight objects, which were in a snapshot, absent from the next one and present in a later one. An interesting issue here is how frequently snapshots have to be taken to reliably sample cache activity.

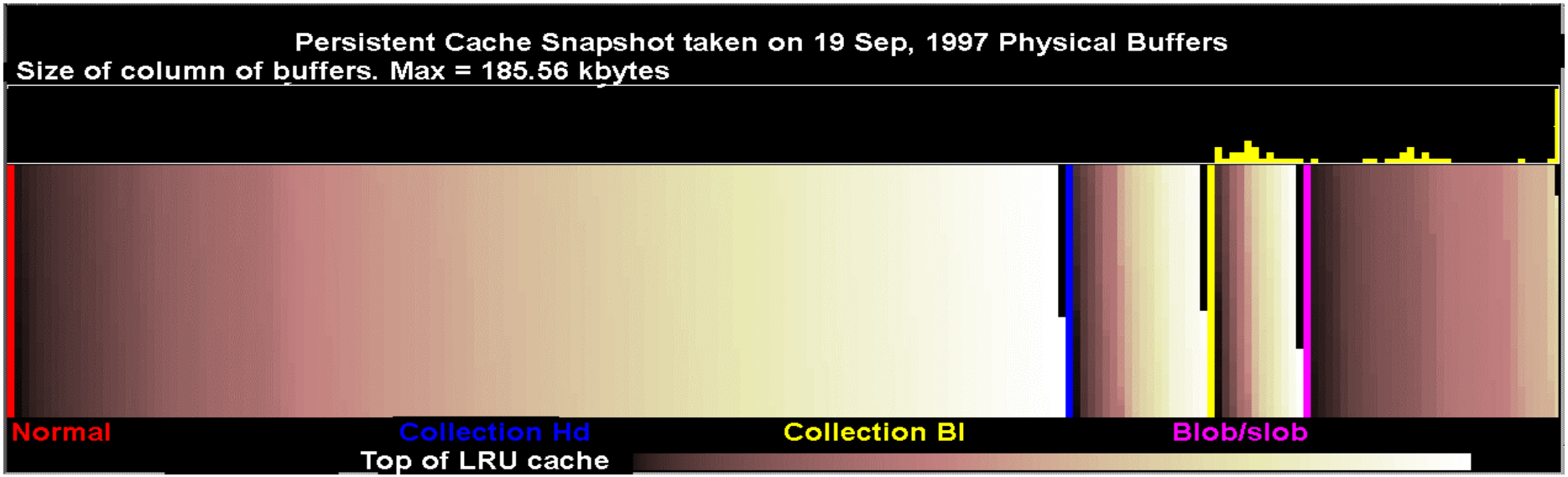

Figure 1 a. Display of the persistent cache from the test snapshot with buffers grouped by their physical type (normal, collection header, collection block, blob/slob).

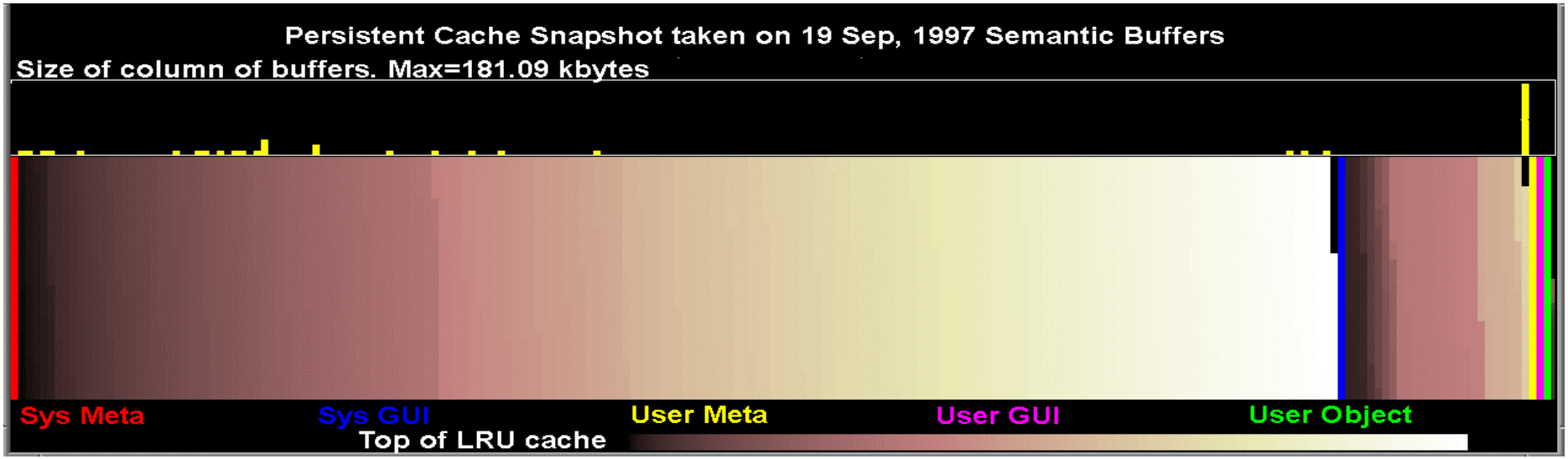

Figure 1 b. Display of the persistent cache from the test snapshot with buffers grouped according to their semantic type (system meta object, system GUI object, user meta object, user GUI object, user object, unknown object).

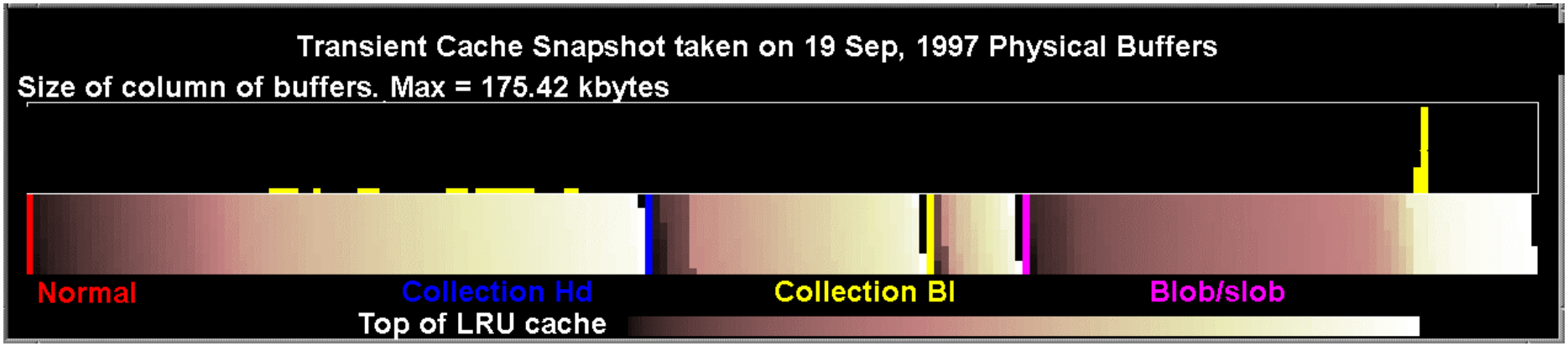

Figure 1 c. Display of the transient cache from the test snapshot with buffers grouped by their physical type (normal, collection header, collection block, blob/slob).

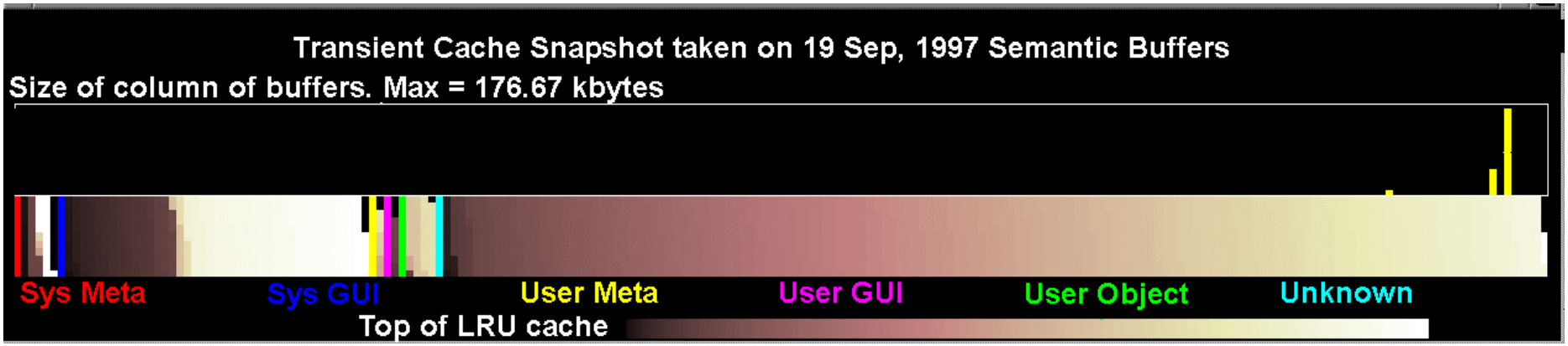

Figure 1 d. Display of the transient cache from the test snapshot with buffers grouped according to their semantic type (system meta object, system GUI object, user meta object, user GUI object, user object, unknown object).

Implementation

The program to create the displays was written in Matlab. It reads the snapshot data and, on the basis of the number of buffers present and the area of the screen the user wishes to use to represent each buffer (typically 2-4 pixels), determines the area of the screen which will be used for the display. A rectangular matrix of the same size in pixels as the area of the screen, which will be used to display it, is used to prepare the information for display. The numerical value placed in each element of the matrix is such that when the matrix is displayed as an image, the colour map translates it to the correct colour.

Conclusions

The visualisation of JADE cache snapshots shown here gives a basic view of the information contained in each snapshot. There are a number of ways in which this could be enhanced to improve the information, which can be accessed, especially for multiple snapshots. We have also investigated the possibility of using 3D graphics to explore aspects such as the distribution of buffers in terms of LRU position and age. All of these approaches need to be applied to a large number of snapshots arising from a variety of JADE applications and operating conditions. Then a better idea of the utility of each approach will be obtained and enhancements can be developed.

Acknowledgements

Most of the work reported here was carried out while Alan McKinnon was a sabbatical visitor at the Sydney University Vislab from July to September 1997. The generous hospitality, access to facilities and technical support of Assoc Prof Bernard Pailthorpe and his staff are gratefully acknowledged.

References

van der Deijl Eric, Kanbier Gerco, Temem Olivier, Granston Elena D. (1997) *A Cache Visualization Tool* IEEE Computer 30(7) p71-78

Stallings W. (1993) *Computer organization and architecture : principles of structure and function,* Macmillan International, NY

Appendix A

Information in the Snapshot Files

Some of the information contained in each snapshot file is listed below. The transient buffers and persistent buffers are assumed to be in separate files although the information is the same in each case.

descriptive header record – this information is shown at the top of the display

cache summary record
- time of snapshot
- number of cache hits
- number of cache misses
- number of hits where the buffer was at the top of the LRU

data records – one for each buffer in the cache at the time of the snapshot
- position in the cache (i.e. LRU order)
- ID for object in buffer
- semantic category (see below)
- physical category (see below)

semantic categories are	1	system meta object
	2	system GUI object
	3	user meta object
	4	user GUI object
	5	user object
	6	unknown object

physical categories are	1	normal object
	2	collection header
	3	collection block
	4	blob/slob