The MaStA I/O Trace Format

S.J.G. Scheuerl†, R.C.H. Connor†, R. Morrison†, J.E.B. Moss¥ & D.S. Munro†

†School of Mathematical and Computational Sciences,
University of St Andrews,
North Haugh, St Andrews, Fife, KY16 9SS, Scotland
Email: {stephan, richard, ron, dave}@dcs.st-and.ac.uk

¥Department of Computer Science, University of Massachusetts,
Amherst, Massachusetts, MA 01003, U.S.A.
Email: moss@cs.umass.edu

1 Introduction

MaStA [SCM+95] is an analytical framework for comparing the I/O costs of database recovery mechanisms under a variety of different workloads and configurations. A major part of the strategy for validating the assumptions of the model involves analysing I/O traces taken from real and simulated database systems running under different workloads. These traces may be used for example to calibrate device simulators, or to compare I/O costs of devices. This document provides a detailed description of the format of the I/O traces.

2 Overview

It has been shown that the cost of recovery mechanisms can be critical to the overall performance of data-intensive applications with I/O bandwidth being a limiting factor. Many recovery mechanisms have been invented, each with different performance tradeoffs [HR83]. Each technique's cost involves not only the overhead of restoring data after failures but also the time and space overhead required to maintain sufficient recovery information during normal operation to ensure recovery. Under different workloads and configurations these crash recovery mechanisms exhibit different costs.

The MaStA I/O cost model [SCM+95] provides an analytical framework for comparing such recovery mechanisms under a variety of different workloads and configurations. It is structured independently of machine architectures and application workloads and determines costs in terms of I/O categories, access patterns and application workload parameters. An outline of the model is illustrated in Figure 1.

Each recovery mechanism's I/O costs are broken down into a number of independent I/O cost categories, such as "data reads" or "commit writes". The total I/O cost of a mechanism is the sum of the cost of each category.

The cost of each category is derived from the number of accesses it incurs of certain access pattern. Access patterns such as sequential, asynchronous, clustered synchronous and unclustered synchronous are assigned to each recovery mechanism on a platform with the knowledge that the patterns have significantly different costs. The number of accesses is calculated from workload parameters, such as page and object size, and the locality of objects accessed. The cost of a category is the product of the number of accesses and the I/O access pattern cost, or a sum of a number of such products.

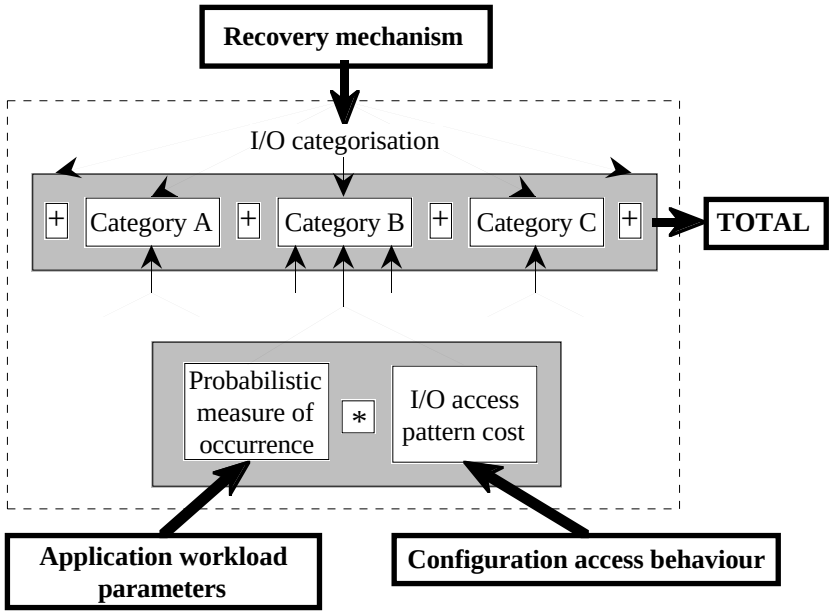


Figure 1: An Overview of MaStA

The model is based on four assumptions:

- The significant extra cost associated with a recovery mechanism is the extra I/O activity generated: the extra CPU costs are not significant.
- The interaction between the different categories and patterns of I/O accesses is not significant.
- The cost of running an I/O stream in each pattern is the same as multiplying the average cost of that pattern by the number of accesses.
- The cost of running the I/O streams generated by an application is approximately the same as running the I/O streams generated by the workload abstraction of the application.

The strategy for validating these assumptions is outlined in Figure 2.

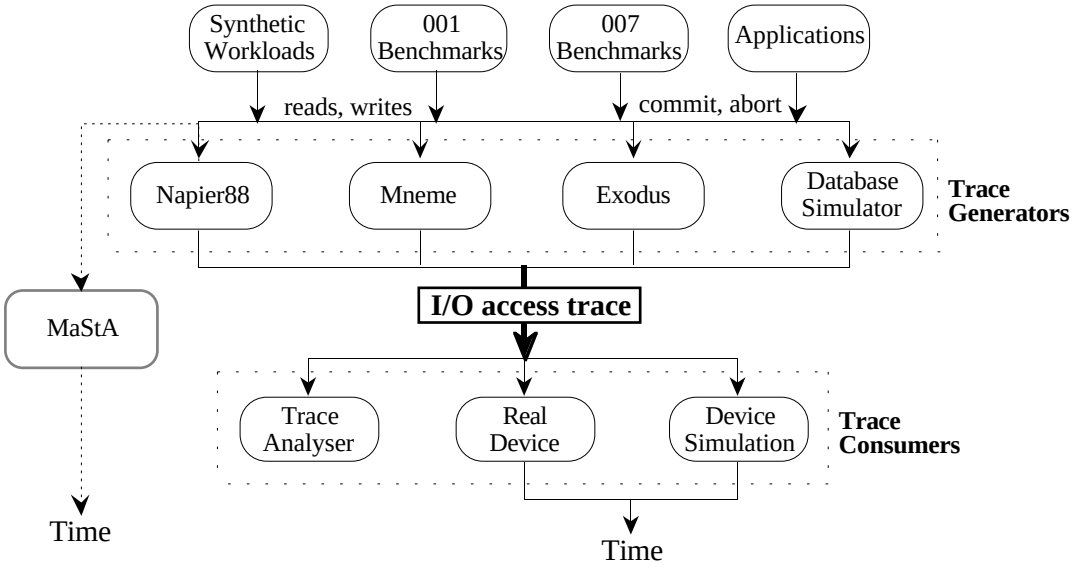


Figure 2: Validation Strategy

Sequences of database accesses generated from synthetic workloads, applications and standard benchmarks 001 [CS92] and 007 [CDN93], are fed into a number of database systems and simulators reflecting different recovery mechanisms and buffer models. Traces recording the generated I/O accesses are taken from these systems. These traces may be used by trace consumers in a number of ways:

- The traces may be analysed and used in the validation of the assumptions of the MaStA model;
- Trace consumers may run the I/O traces over real and simulated devices in order to calibrate the device simulators against the real devices;
- Traces may be run over devices to compare the I/O costs.

3 I/O Traces

A trace is a file containing a sequence of I/O trace entries recording read, write and synchronisation (sync) operations performed on a database. I/O trace entries are interspersed with configuration entries recording additional information which can be used by trace consumers to configure devices. This additional information includes text describing the system configuration and timings to allow CPU and I/O costs to be derived.

The trace format allows *regions* to be defined in a trace to distinguish between different logical areas of storage used by the database system. For example in a logging mechanism two regions are defined for the log and the database area. The trace generator specifies whether these regions are bound to the same device or to different devices. Figure 3 illustrates two systems, one in which both regions used by the system are bound to a single device, and another in which the two regions are bound to different devices.

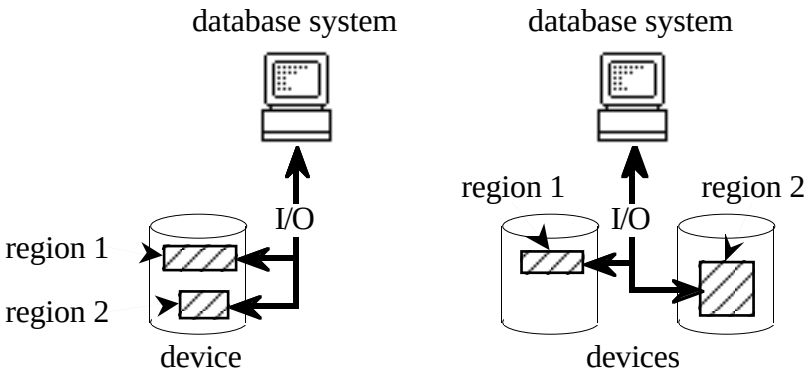


Figure 3: Mapping Regions to Devices

Reads and writes are recorded in a trace as operating over one or more blocks of a particular region. Synchronisation operations are recorded as operating over one or more regions.

The trace format allows *streams* to be defined to reflect some notion of concurrency, each stream representing a concurrent action. Each read or write is recorded with respect to a stream, allowing trace consumers to identify and follow threads of I/O accesses generated by a particular action.

The trace format permits trace entries to record the predicted I/O access cost of a read or write, and to record the I/O category in which the operation is performed. The

predicted I/O costs may be compared with the costs obtained from analysing the trace or running the trace over devices.

4 Trace Format

This section contains descriptions of the different forms of trace entries. The descriptions are intended for use by database systems programmers to provide a standard format for I/O traces. The descriptions are also intended to allow trace consumers to interpret these traces. A sample trace is given in the Appendix.

Section 4.1 describes I/O trace entries and Section 4.2 describes configuration trace entries. Below the name of an entry, the format shows how the entry is encoded. The description provides information about the use of the entry, and a description of the individual entry fields.

Each trace entry is encoded into two 32-bit words, word_0 and word_1 (Figure 4). The bytes of an entry are numbered left to right as bytes 0 to 7. When an entry records an integer, the integer is held in bigendian format. Unless otherwise stated, byte and integer values are unsigned.

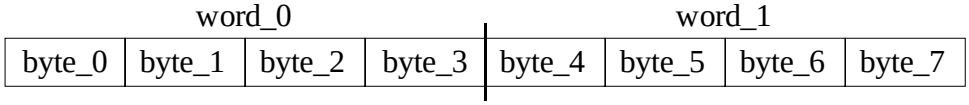


Figure 4: Byte Ordering of a Trace Entry

4.1 I/O Trace Entries

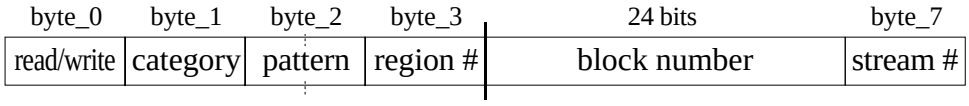
I/O trace entries are used to record read, write and synchronisation operations in a trace. The particular operation recorded is indicated by the value held in byte_0:

1 read
2 write
3 sync
4 block operation

The values 5 to 255 are reserved for future use. The value 0 indicates a configuration entry described in Section 4.2.

4.1.1 Read and Write Trace Entries

Format:



Description:

Read and write trace entries are used to record read and write operations. If no concurrency is reflected in a trace, the stream number is 0.

read/write

byte_0 specifies whether the operation is a read indicated by the value 0, or a write indicated by the value 1.

category

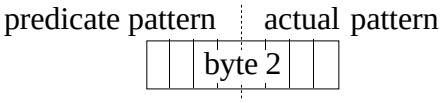
byte_1 records the I/O category in which the read or write is performed. The categories and their representative values are:

0 Data
1 Recovery
2 Installation
3 Commit

Values 4 to 255 are reserved for future use.

pattern

The upper 4 bits of byte_2 hold the predicted access pattern incurred by the operation.



The lower 4 bits of byte_2 are reserved for use by trace consumers to record the actual access pattern incurred. The upper four bits correspond to an I/O pattern:

0 not predicted
1 sequential
2 asynchronous
3 clustered synchronous
4 unclustered synchronous

The value 0 indicates that the access pattern was not predicted when the trace entry was recorded. Values 5 to 15 are reserved for future use.

region number

byte_3 records the region number on which the read or write operates.

blocks number

byte_4 to byte_6 record a 24-bit integer specifying the block as an offset over which the read or write operates.

stream number

byte_7 records a stream number from 0 to 255.

4.1.2 Sync Trace Entries

Format:

byte_0	byte_1	byte_2	byte_3	word_1
3	-	-	region	region bitmap

Description:

Sync trace entries record synchronisation operations over one or more regions. The values of byte_1 and byte_2 are undefined.

region

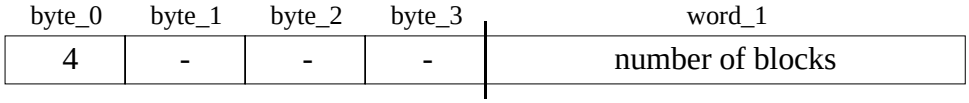
byte_3 records a signed value between 0 and 31 specifying the particular region synchronised. The value -1 in byte_3 indicates that all regions are synchronised. The value -2 in byte_3 indicates that word_1 specifies the regions synchronised.

region bitmap

The value -1 in byte_3 indicates that word_1 contains a bitmap specifying the regions synchronised. The least significant bit of word_1 corresponds to region 0 and the most significant bit corresponds to region 31. A value other than -1 in byte_3 indicates that word_1 is undefined.

4.1.3 Block Operation Entries

Format:



Description:

If more than one contiguous block is operated over by a read or write a block operation entry records the number of blocks. The read or write is recorded immediately after the block operation entry by another I/O trace entry. The read or write trace entry specifies the first block operated over. The values of byte_1 to byte_3 of the block operation entry are undefined.

number of blocks

word_1 records the number of contiguous blocks over which the read or write is performed.

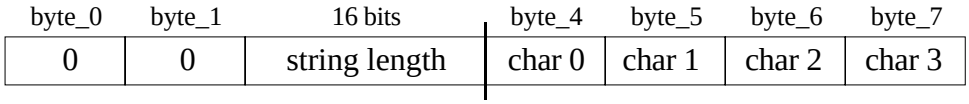
4.2 Configuration Entries

Configuration entries are used to record additional information in a trace. These entries are indicated by the value 0 in byte_0. The particular form of a configuration entry is indicated by the value held in byte_1:

0 string header entry
1 region mapping entry
2 region size entry
3 block size entry
4 time entry
5 format version entry
6 end of trace entry

4.2.1 String Header Entries

Format:



Description:

String header entries are used to record the start of a string in a trace. Each string is recorded by a string header entry followed by a sufficient number of contiguous string entries to hold any remaining characters. The format of a string entry is described in Section 4.2.2.

string length

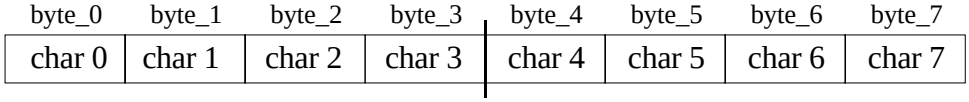
byte_2 and byte_3 hold a value up to 65535 specifying the total number of characters in the string, including the terminating null character.

char 0 -> char 3

byte_4 to byte_7 record the first 4 characters of the string. All characters are recorded in ASCII format.

4.2.2 String Entries

Format:



Description:

String entries follow contiguously after a string header entry to record the remaining characters of a string. A string is of the form “X=Y” where X is a variable name and Y is a value associated with the variable. Strings are null terminated. The following variables are given here as a guide:

DATE=	date in the format YYMMDD
TIME=	24 hour time in the format HHMM[SS]
APPL=	name of the application or benchmark
USER=	the user’s name
EMAIL=	the user’s email address
HW-CPU=	CPU specifications
HW-CACHE=	cache specifications
HW-MEM=	quantity of physical memory
HW_DEV0=	make of device number 0
HW_DEV1=	make of device number 1
HW_DEV2=	make of device number 2
SW-OS=	operating system and version
SW-DB=	database system
SW-RM=	recovery mechanism

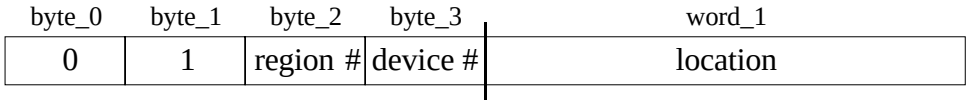
Producers of traces may devise other variables. The string describing a device is recorded before recording the first region mapping entry referring to the device.

char 0 -> char 7

All bytes of the entry are used to record characters.

4.2.3 Region Mapping Entry

Format:



Description:

A region number is mapped to a device number using a region mapping entry. Up to 32 regions may be mapped. A region must be mapped to a device before recording the first I/O trace referring to the region. A region may be re-mapped to a different device at any time using another region mapping entry.

region number

A value between 0 and 31 in byte_2 specifies the region number.

device number

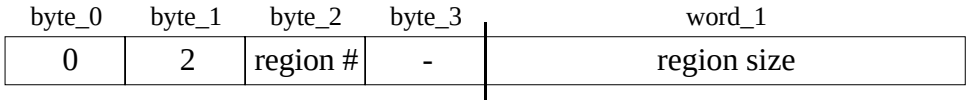
byte_3 specifies the device number being mapped. The device number corresponds to a device described in a string such as ‘HW_DEV0=CDC Wren V SCSI drive‘ recorded earlier in the trace.

location

word_1 records the location on the device to which the start of the region is mapped. The location is given as a offset in blocks from the start of the device.

4.2.4 Region Size Entry

Format:



Description:

A region size entry is used to bound the size of a region. The value of byte_3 is undefined. The size of a region is recorded after the region has been mapped to a device using a region mapping entry. If no region size entry is recorded for a particular region it is assumed that the upper bound of the region was unknown during tracing. Note that the upper bound of a region may be determined by the trace consumer by scanning the region and block numbers recorded in I/O trace entries.

region number

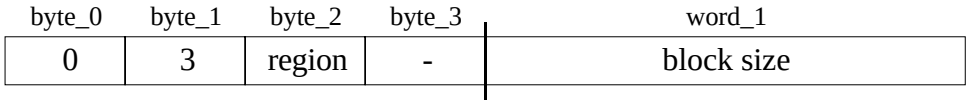
byte_2 specifies the region number being mapped.

region size

The size of the region as a number of blocks is recorded in word_1.

4.2.5 Block Size Entries

Format:



Description:

Block size entries record the block size used by the database system over the regions it uses. A block size entry may refer to one region or to all regions. The value of byte_3 is undefined.

The block size of an entry is recorded after the region has been mapped to a device using a region mapping entry. The block size entry for a given region is recorded before recording the first I/O trace entry referring to the region. If the block size of a region is not specified it is assumed that the block size of the region is 4096 bytes. The block size of a region may be changed during a trace using another block size entry.

region

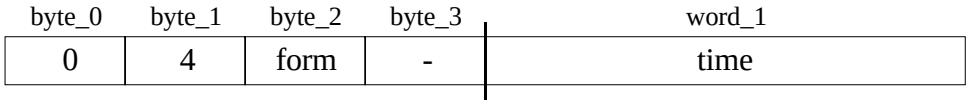
An unsigned value between 0 and 31 in byte_3 specifies the region number. The value -1 in byte_3 indicates that the block size refers to all regions.

block size

The block size is recorded as a number of bytes in word_1.

4.2.6 Time Entries

Format:



Description:

Time entries are used to record two forms of time: real time, and elapsed time since recording the previous elapsed time entry. The value of byte_3 is undefined. Elapsed time entries are recorded immediately before and after recording I/O trace entries, allowing CPU and I/O costs of the database system to be isolated. Elapsed time entries are optional.

form

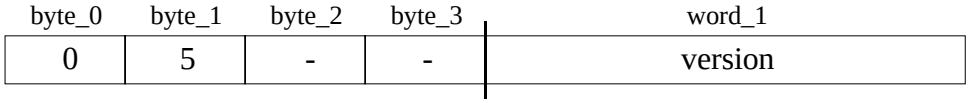
The specific form of a time entry is indicated by byte_2. The value 0 indicates that the time entry records real time. The values 1 and 2 in byte_2 indicate that the entry records elapsed time in microseconds since the previous elapsed time entry. The value 1 indicates that the entry records elapsed time immediately before performing an I/O. The value 2 indicates that the entry records elapsed time immediately after performing an I/O.

time

When byte_2 holds the value 0, word_1 records an integer denoting the real time in seconds since midnight 1st January 1970. Otherwise, word_1 records an integer specifying elapsed CPU time in microseconds since the last elapsed time record.

4.2.7 Format Version Entries

Format



Description

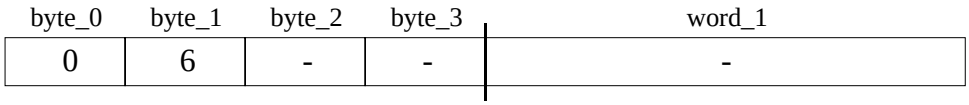
A format version entry specifies the trace format version used to produce the trace. The first entry in a trace must be a trace format version entry. The values of byte_2 and byte_3 are undefined.

version

word_1 records an integer specifying the trace format version number.

4.2.8 End Trace Entries

Format:



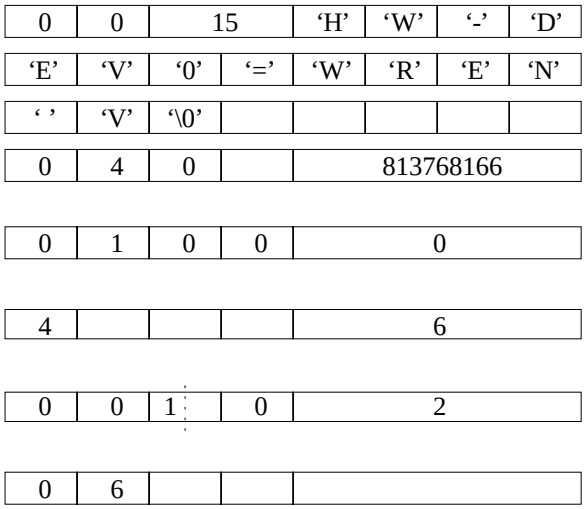
Description:

An end of trace entry records the termination of a trace. The values of byte_2, byte_3 and word_1 are undefined.

5 Appendix

The following is a sample trace which may be used by a trace consumer. All values are displayed in decimal.

Trace Entries								Description
0	5			0				trace version number 0
0	0	17		‘N’	‘A’	‘M’	‘E’	string header entry and string entries containing “NAME=S. Scheuerl”
‘=’	‘S’	‘.’	‘ ’	‘S’	‘c’	‘h’	‘e’	
‘u’	‘e’	‘r’	‘l’	‘\0’				
0	0	20		‘S’	‘W’	‘-’	‘D’	string header entry and string entries containing “SW-DB=Napier88 V2.0”
‘B’	‘=’	‘N’	‘a’	‘p’	‘i’	‘e’	‘r’	
‘8’	‘8’	‘ ’	‘V’	‘2’	‘.’	‘0’	‘\0’	



6 References

[CDN93] Carey, M.J., DeWitt, D.J. & Naughton, J.F. "The OO7 Benchmark". In SIGMOD Conference on the Management of Data, 1993.

[CS92] Cattell, R.G.G. & Skeen, J. "Object Operations Benchmark". ACM Transactions on Database Systems 17,1 (1992) pp 1-31

[HR83] Haerder T. & Reuter, A. "Principles of Transaction-Oriented Database Recovery". Computing Surveys, Vol. 15, No. 4, Dec. 1983, Pages 287-317.

[SCM+95] Scheuerl, S.J.G., Connor, R.C.H., Morrison, R., Moss, J.E.B. & Munro, D.S. "The MaStA I/O Cost Model and its Validation Strategy". In the Proceedings of the Second International Workshop on Advances in Databases and Information Systems (ADBIS'95), Moscow, June 27-30 1995, Volume 1, pp 165-175.