Validating the MaStA I/O Cost Model for DB Crash Recovery Mechanisms

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

†School of Mathematical and Computational Sciences,
University of St Andrews,
North Haugh, St Andrews, Fife, KY16 9SS, Scotland

¥Department of Computer Science, University of Massachusetts,
Amherst, Massachusetts, MA 01003, U.S.A.

Abstract

The design of database crash recovery mechanisms on modern computer systems must take account of the relative speed of processors over disks. In these systems disk I/O activity is the dominant expense and the disk transfer time relative to seek time makes patterns of disk access significant. The MaStA (**Ma**ssachusetts **St A**ndrews) cost model for database crash recovery mechanisms, designed by the authors, provides finer grained distinctions of I/O costs than previous work by being structured independently of machine architectures and application workloads and refining costs in terms of I/O categories, access patterns and application workload parameters. The main features of the model are:

- Cost is based upon a probabilistic estimation of disk activity, broken down into sequential, asynchronous, clustered synchronous, and unclustered synchronous disk accesses for each recovery scheme.
- The model is calibrated by different disk performance characteristics, either measured by experiment or predicted by analysis.
- The model may be used over a wide variety of workloads, including those typical of object-oriented and database programming systems.

The MaStA model is based on the assertion that four basic assumptions hold true. This paper briefly describes the MaStA model and outlines the validation strategy undertaken to confirm these assumptions.

1 The MaStA Cost Model

The cost of recovery mechanisms can be critical to the overall performance of data-intensive applications with I/O bandwidth being a limiting factor. Hence 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.

Modern trends in hardware design have given a disproportionate improvement in processor speed compared to disk access time, and within disks themselves a disproportionate improvement in transfer rates compared to seek times. These factors change the engineering tradeoffs upon which recovery mechanisms are based, and recently designed mechanisms tend to favour extra processor activity to reduce disk I/O, and more asynchronous (scheduled) I/O as opposed to synchronous access [EB84, OS94]. In addition, one type of I/O access can be significantly faster than other types of access depending on the disk and operating system factors. In [SCM+95a] the ratio of the cost of unclustered synchronous writes to sequential reads was observed to be a factor of six in a measured system. The advent of object-oriented database systems, and persistent and database programming languages has also changed the demands made upon recovery mechanisms in the way that they access data, meta-data and programs. The purpose of the model outlined here is to provide an analytical framework for comparing recovery mechanisms under a variety of different workloads and configurations.

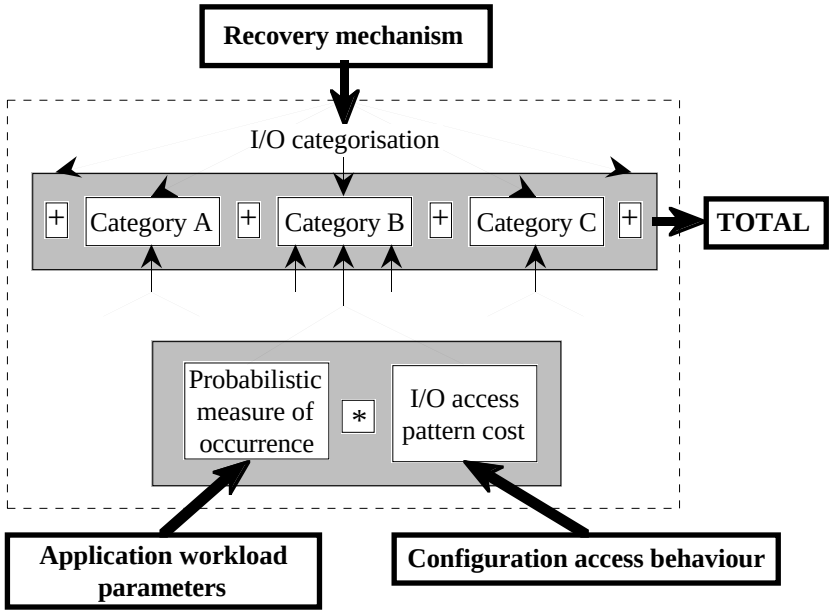

Figure 1: An overview of MaStA

Figure 1 shows a simplified view of the MaStA model. Each recovery mechanism's cost is broken down into constituent independent I/O cost categories, such as “data reads” or “commit writes”. The overall cost of a mechanism is the sum of the costs of each category:

$$\text{Total Cost} = \sum \text{CatCost}(i),\ (i \in \text{Categories})$$

Each category's cost is derived from the number of accesses, n, it incurs of each access pattern, A :

$$\text{CatCost}(i) = \sum n_j * A_j,\ (j \in \text{Access Patterns})$$

For any given medium, a set of access patterns is developed such that each pattern is believed to have significantly different costs. For example, sequential writes on disk systems are usually faster than unclustered synchronous writes. The number of accesses of each pattern is derived from workload parameters, such as page and object size, and the density of objects within pages.

The derivation of a cost estimate for a particular combination of mechanism, configuration and workload is derived by analysing:

- The mechanism: identifying the cost categories, and for each category the access patterns and number of accesses. The crucial refinement of the MaStA model is to distinguish various I/O access patterns, on the basis of their significantly different costs.

- The configuration: determining the average cost of each access pattern experimentally or analytically.

- The workload: measuring and choosing values for the workload parameters which determine the number of I/Os in each category.

The identification of these three categories allows the MaStA model to encompass the patterns of usage in both traditional and modern database systems. In [SCM+95a, SCM+95b] the MaStA model is described in greater depth together with an analysis and comparison of four recovery mechanisms under differing workloads.

2 Validation

Three major abstractions are made to simplify the process of making a cost estimate, based upon the following critical underlying assumptions:

Recovery mechanism abstraction

Each recovery mechanism is analysed to assess its I/O costs in a number of different categories. The total cost derived by the model is the sum of these categories. The purpose of the categorisation is to simplify the analysis of the recovery mechanism, and indeed it has proved relatively straightforward to perform such analysis. The success of this abstraction depends heavily upon two assumptions:

Assumption 1: The significant extra cost associated with a recovery mechanism is in extra I/O activity generated: extra CPU costs incurred by recovery mechanisms are not significant.

Assumption 2: The interaction between the different categories and patterns of I/O accesses is not significant.

Disk performance abstraction

The performance of I/O devices such as disks is highly dependent upon the ordering and synchronicity attributes of the access streams. The cost abstraction assigns an average cost per disk access to each of the defined patterns. The average cost may be obtained by simulation, experiment or further analysis of the device in question.

Assumption 3: The cost of running an I/O stream in a given pattern is the same as multiplying the predicted average cost of that pattern by the number of accesses.

Workload abstraction

The last abstraction is over the workload associated with the application. As the interest is only in I/O behaviour, this need not encompass any CPU activity of the application, but only in the manner in which it accesses objects. The application is characterised in terms of factors such as average object size, object density, update density. Again, a major issue is to keep the characterisation sufficiently simple that it may be measured or estimated for a given application workload.

Assumption 4: The cost of running the I/O stream generated by an application is approximately the same as running the I/O stream generated by the workload abstraction of the application.

Once the MaStA model has been calibrated there is a final assumption that is used in estimating the cost of any system. The assumption is that there are no significant phase changes in the performance of the system [ABJ+92].

2.1 Validation Strategy

The strategy for validation of the MaStA model is outlined in Figure 2. A main feature of this approach is that results from simulations are compared alongside experimental measurements from real systems at the transaction workload, database and device levels. Validation is achieved through an analysis of the resulting times and traces from a variety of combinations of synthetic and measured experiments.

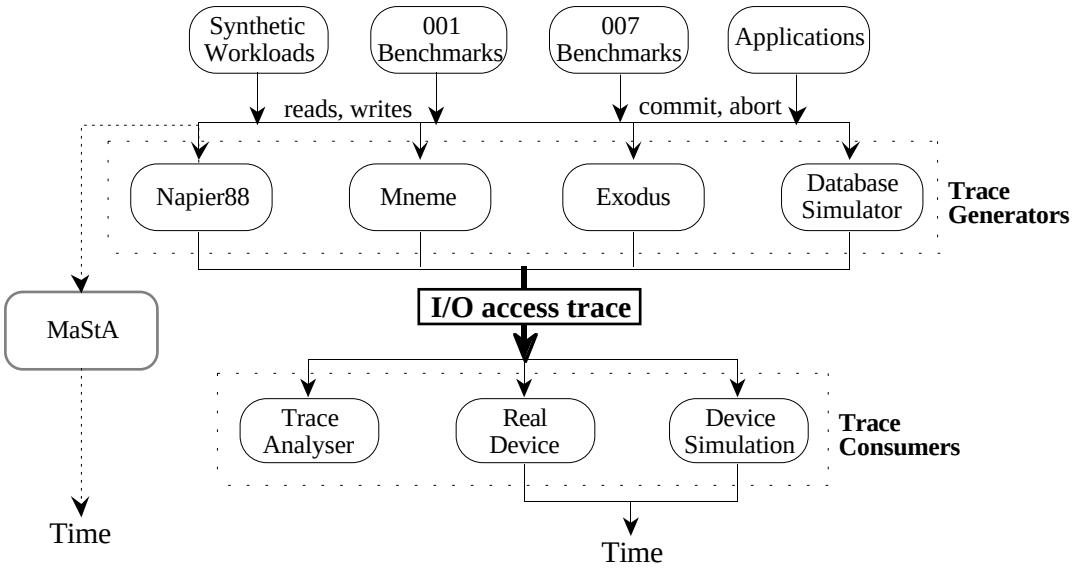

Figure 2: Validation Strategy

A trace of database accesses generated from synthetic workloads and the standard 001 [CS92] and 007 [CDN93] benchmark suites or other applications are fed into a number of database systems and simulators reflecting different recovery methods and buffer models. An I/O access stream is produced from these systems tagged by I/O cost category. These traces are then run across a number of real and simulated devices to produce an overall cost. The Napier88 systems [MBC+89] are supported by object stores [BDM+90, MCM+94] which utilise a before-image and an after-image shadow-paging scheme whilst Mneme [MS88] and Exodus [CDF+86] use page and object-based logging respectively. By feeding into these systems representative samples of applications, a range of performance of recovery methods is obtained.

2.2 Validation Status

At the time of writing, test runs and analysis of experimental data are at an early stage with the main effort concentrated on the definition of a suitable I/O trace format [SCM+95c] and on the validation of assumptions 2 and 3. The trace identifies the sequence of blocks read and written from a particular application tagged by its cost category. I/O tracing has been incorporated into Napier88 systems running Datasafe [SCM+95d], a hybrid of the DBCache [EB84] recovery scheme and Flask [MCM+94], an after-image shadow paging [Lor77, Cha78] mechanism. Traces generated so far record the I/O activity from the OO1 benchmark suites as well as “real” workloads.

A number of these traces have been run over real disks and an in-house disk simulator. One early observation is that the combination of read/write buffering in real disks and systems has a very significant impact on performance indicating the need for a fair level of sophistication in any disk simulator. These traces were run on Sun SparcStations and a DEC Alphas using CDC WrenV and WrenVII disks. The results were measured over raw and "cooked" partitions on these drives and on single files with and without the sticky bit set. (The sticky bit means that the system's page cache will not be used to hold the file's data and that the file's inode modification times may not necessarily be correctly recorded on permanent storage.) As expected the runs produced different results according to the characteristics of the file types. However it is clear from the results that the average costs of I/Os for the different access patterns is significant in reality and hence should be accounted for in any cost model.

In addition a trace analyser has been designed which may be used to determine if the predicted behaviour of a particular recovery mechanism matches the actual I/O patterns incurred. The eventual intention of the MaStA model is that it will be able to accurately predict the running time of an arbitrary mixture of application, recovery mechanism and platform.

A document describing the I/O trace format together with sample traces, analyser programs and results can be accessed from http://www-fide.dcs.st-and.ac.uk.

3 Acknowledgements

This work was undertaken during the visits of Eliot Moss to the University of St Andrews as an EPSRC Senior Visiting Fellow on grant GR/K34924 and GR/K55509 and National Science Foundation grant CCR-9211272.

4 References

[ABJ+92] Atkinson, M.P., Birnie, A., Jackson, N. & Philbrow, P.C. "Measuring Persistent Object Systems" In Proc. 5th International Workshop on Persistent Object Systems, San Miniato, Italy (1992). In Persistent Object Systems (Eds. A.Albano & R.Morrison). Springer-Verlag pp 63-85.

[BDM+90] Brown, A.L., Dearle, A., Morrison, R., Munro, D.S. & Rosenberg, J. "A Layered Persistent Architecture for Napier88". International Workshop on Computer Architectures to Support Security and Persistence of Information, Universität Bremen, West Germany, (May 1990). In Security and Persistence. (Eds. J.Rosenberg & L.Keedy). Springer-Verlag, 155-172.

[CDF+86] Carey, M.J., DeWitt, D.J., Frank, D., Graefe, G., Muralikrishna, M., Richardson, J.E. & Shekita, E.J. "The Architecture of the EXODUS Extensible DBMS". In Twelfth International Conference on Very Large Data Bases, 1986 pp 52-65.

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

[Cha78] Challis, M.F. "Database Consistency and Integrity in a Multi-user Environment". Databases: Improving Useability and Responsiveness (1978) pp 245-270.

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

[EB84] Elhardt, K. & Bayer, R. "A Database Cache for High Performance and Fast Restart in Database Systems". ACM Transactions on Database Systems, 9,4 (1984) pp 503-525.

[HR83] Haerder, T. & Reuter, A. “Principles of Transaction-Oriented Database Systems”. ACM Computing Surveys, 15,4 (1983) pp 287-318.

[Lor77] Lorie, A.L. “Physical Integrity in a Large Segmented Database”. ACM Transactions on Database Systems, 2,1 (1977) pp 91-104.

[MBC+89] Morrison, R., Brown, A.L., Connor, R.C.H. & Dearle, A. “The Napier88 Reference Manual”. University of St Andrews Technical Report PPRR-77-89 (1989).

[MCM+94] Munro, D.S., Connor R.C.H., Morrison, R., Scheuerl, S. & Stemple, D.W. "Flask - A Flexible Layered Architecture for Supporting Concurrency Control Schemes". To appear in proceedings of the 6th International Workshop on Persistent Object Systems, Tarascon, France (September 1994)

[MS88] Moss, J.E.B. & Sinofsky, S. “Managing persistent data with Mneme: Designing a reliable shared object interface”. In Dittrich, K.R. (ed.) Advances in Object-Oriented Database Systems: Second International Workshop on Object-Oriented Database Systems, LNCS 334, Springer-Verlag, 1988 pp 298-316.

[OS94] O'Toole, J. & Shrira, L. "Opportunistic Log: Efficient Installation Reads in a Reliable Object Server". Technical Report MIT/LCS-TM-506, March 1994. To appear in 1st International Symposium on Operating Systems Design and Implementation, Monterey, CA (1994).

[SCM+95a] Scheuerl, S., Connor R.C.H., Morrison, R., Moss, J.E.B. & Munro, D.S. “MaStA - An I/O Cost Model for Database Crash Recovery Mechanisms” Technical Report CS/95/1 (1995), University of St Andrews.

[SCM+95b] 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.

[SCM+95c] Scheuerl, S.J.G., Connor, R.C.H., Morrison, R., Moss, J.E.B. & Munro, D.S. "The MaStA I/O trace Format". Technical Report CS/95/4 (1995), University of St Andrews.

[SCM+95d] Scheuerl, S.J.G., Connor, R.C.H., Morrison, R., Munro, D.S. & Moss, J.E.B. "The DataSafe Failure Recovery Mechanism in the Flask Architecture". Submitted to ACSC'96 (Melbourne).