A Brief Tutorial on Epsilon Serializability and a Survey of Recent Work

Calton Pu[1]
Department of Computer Science and Engineering
Oregon Graduate Institute
P.O. Box 91000
Portland, OR 97291-1000

Internet: calton@cse.ogi.edu

Technical Report No. OGI-CSE-93-008

Abstract

Epsilon Serializability (ESR) is a generalization of classic serializability (SR). ESR supports more concurrency and autonomy than SR by allowing a bounded amount of inconsistency in transactions that can tolerate it. ESR has three main advantages over previous "weak consistency" models: (1) ESR is a general framework, applicable to a wide range of application semantics; (2) ESR is upward-compatible, since it reduces to SR as $\epsilon \rightarrow 0$; and (3) ESR has number of efficient algorithms that support it. Since its introduction [17], several papers have been published on ESR describing its properties, applications, algorithms, and performance evaluation. We briefly describe the concept of ESR, summarize the published work on ESR as well as ongoing research, and outline the future research directions.

1 ESR Basics

An important assumption in ESR is the existence of a distance function and an associated regular geometry in the database state space. Fortunately, many of the real world database state spaces fulfill this requirement. For example, integers and real numbers in banking, airline, and scientific data are cartesian spaces that have a natural definition of distance function (the difference) and the regular geometry of a metric space [11]. Efficient algorithms that support ESR use these properties to preserve the inconsistency bounds. The management of inconsistency in database state spaces with irregular geometries remains an interesting area of research.

Another dimension in the ESR research is the algebraic properties of transaction operators. A classic transaction is defined as a sequence of operations that transform a consistent database state into another consistent state. This definition bypasses the issue of *what* operators are employed within a transaction. In ESR work, a bounded amount of inconsistency may be introduced into a transaction and we are interested in how this inconsistency is propagated within the transaction as well as in the database. This line of investigation [21, 4] benefits from previous work on operation semantics (e.g., [2], cite UCSB work as a potential example). The results will help application designers in estimating the amount of inconsistency tolerated by the applications. Unlike the previous work on operation semantics, however, our algorithms do not depend on particular operation semantics. Therefore, we have database state space geometric properties and ESR algorithms on the system side, contrasting with transaction operator algebraic properties on the application side.

The application interface to ESR systems is an extension of classic transactions, called Epsilon-Transaction (ET). Each ET may specify the bound on the amount of inconsistency imported into it

[1]This research is partially supported by Oki Electric and NSF.

Trans-ϵ-spec	*import-limit* $= 0$	*import-limit* > 0
export-limit $= 0$	Transaction	Q^{ET}
export-limit > 0	U^{ET}	G^{ET}

Table 1: Three Kinds of ETs

and the amount of inconsistency exported by it, called *import-limit* and *export-limit*, respectively. Together, *import-limit* and *export-limit* are called Transaction Epsilon Specification, or trans-ϵ-spec. In addition, data items that may include some inconsistency in them can be specified with data-ϵ-spec. Typically, these data items are denoted by an interval of possible values or a central value with plus and minus interval around it.

An extended TP system guarantees ESR execution in a manner transparent to ET programmers. In the simplest case, (*import-limit* $= 0$ and *export-limit* $= 0$) the ET reduces to an SR transaction. At the next level of generality, an ET has either *import-limit* > 0 or *export-limit* > 0, but not both. In Table 1 we see either queries (Q^{ET}) with *import-limit* > 0 or updates (u^{ET}) with *export-limit* > 0. In this case, the history of database update operations (deleting the queries) is SR since the updates do not read in any inconsistent data. But Q^{ET} are allowed to see inconsistent states produced by the updates. Algorithms that maintain Q^{ET} within their *import-limit* and U^{ET} within their *export-limit* are called *divergence control*. In the most general case of a general ET (G^{ET}), where both *import-limit* > 0 and *export-limit* > 0, permanent inconsistency may be introduced into the database and the database content may degenerate with unbounded inconsistency. The data-ϵ-spec bounds the amount of inconsistency for data items. Algorithms that bring the database back to a consistent state are called *consistency restoration*.

2 Published and Ongoing ESR Work

2.1 Algorithm Design

For the restricted model of ESR, consisting only of Q^{ET} and U^{ET}, Wu et al [24] have described a methodology to extend classic conflict-based concurrency control methods such as two-phase locking into a divergence control algorithm that guarantees ESR. The main idea of the methodology is that in every concurrency control algorithm, there is a place where all the potentially non-SR conflicts are detected. In two-phase locking, for example, it is the moment lock conflicts are discovered. In timestamp-based concurrency control, it is the moment read and write requests are checked on their serialization time. In optimistic concurrency control, it is in the serialization graph testing. A divergence control algorithm extends a concurrency control algorithm by allowing a bounded amount of inconsistency at these places.

Informally, a divergence control algorithm is the same as its originating concurrency control algorithm, except for the place where potentially non-SR conflicts are detected. When this detection is triggered in the algorithm, instead of blocking (as in locking) or aborting (as in serialization graph testing), the divergence control accumulates the amount of potential inconsistency (defined by the distance between the old value and the new value of the update involved in the conflict). If the accumulated inconsistency is less or equal than the amount specified by trans-ϵ-spec, then the conflict is allowed. Otherwise, the ET is blocked (in locking) or aborted (in serialization graph testing).

This methodology [24] generates centralized divergence control algorithms directly. We have also applied it to design distributed divergence control algorithms [16]. The extension from centralized to distributed algorithms includes two additional steps. First, we need to distribute the trans-ϵ-spec from the global ET to sub-ETs running at each site. We use the demarcation protocol [3] to do this. The second step is the detection of global conflicts, an additional task due to serializability being a global property. We use the Superdatabase architecture [20] to do this. The result is a family of distributed divergence control algorithms composed of three components: demarcation protocol, local divergence control, and global conflict detection.

While divergence control methods can handle Q^{ET}s and U^{ET}s efficiently, some important applications such as autonomous transaction processing require G^{ET}s. The main problem here is that the database state may become increasingly inconsistent due to the execution of G^{ET}s. Consistency restoration algorithms are extensions of crash recovery algorithms. We have outlined some ideas for consistency restoration in an early paper [18] and described the problem and solutions in more detail in a recent paper [9].

Consistency restoration methods [9] can be divided into three families: undo/redo, compensation-based, and independent updates, in a decreasing ordering of both cost and applicability range. Undo/redo algorithms are similar to classic recovery algorithms, can be applied to most situations, and are the most expensive. Compensation-based algorithms depend on nice properties of compensations (such as commutativity), but are much cheaper. Independent updates are applicable only when an independent source of consistent data exists, and do not require any record keeping at the destination sites.

2.2 Algorithm Evaluation

Kamath and Ramamrithan [10] have refined the specification of inconsistency into a hierarchy. Besides designing a timestamp-based divergence control algorithm to support such hierarchically specified inconsistency bounds, they also evaluated the performance of such an algorithm in a prototype implementation running on DECstations. Their results show consistent and significant increases in TP system concurrency (and decreases in abort rate) when the epsilon bound is increased. This work strengthened the preliminary evaluation done on the original divergence control algorithms [24].

A group at Columbia is evaluating the performance of centralized divergence control algorithms (two-phase locking and optimistic validation) and divergence control algorithms using simulation. (The software is based on the CSIM package.) This model extends a traditional concurrency control evaluation [1]. In fact, the results from [1] are used for validation and as the baseline case when $\epsilon = 0$. The model allows a wide variation of system parameters such as the number of CPUs and disks, and TP parameters such as data access ratio, skew, length, and trans-ϵ-spec. Extensive data is being collected on the sensitivity of throughput and response-time with respect to these system and TP parameters. Preliminary data show that data contention is a significant bottleneck for high performance systems and also a limited trans-ϵ-spec can alleviate this bottleneck.

2.3 Implementation

One of the salient features of epsilon serializability is its compatibility with existing systems and applications. Maintaining this compatibility has been a major goal of ESR work. One major advantage of this compatibility is the ease for the implementation of ESR. Instead of starting from scratch, we can extend an existing TP system based on classic serializability. Concretely, we are implementing the divergence control algorithms on Encina, a commercial TP monitor distributed by Transarc.

Our goals are: (1) to demonstrate ESR as a practical correctness criterion, (2) to evaluate ESR algorithms in a real environment, and (3) to facilitate eventual technology transfer to real

users. This implementation starts with the centralized two-phase locking divergence control [7], as part of an ongoing PhD dissertation work by Shu-Wie Chen [6]. The centralized divergence control implementation changes two modules in Encina: the TP monitor is extended to accept trans-ϵ-spec and the lock manager is extended to accumulate potential inconsistency when R/W conflicts occur.

We plan to extend this implementation to support distributed divergence control. The main components of distributed divergence control are: the centralized divergence control, an implementation of demarcation protocol done at Columbia, and modified Superdatabase global serializability testing routines [14].

2.4 Formalization

A formal characterization of ESR [21] has been done using the ACTA framework. In ACTA, an SR history is defined as a sequence of operations with an empty transitive closure of transaction dependencies. This is equivalent to the Serializability Theorem [5] that uses acyclic transaction dependency graphs. The definition of ESR in ACTA is an extension of the above. An epsilon-serializable history is a sequence of operations with an empty transitive closure of epsilon-transaction dependencies, which are the normal dependencies excluding the conflicts allowed under the trans-ϵ-spec. In the paper we also discuss the accumulation and transformation of inconsistency within ETs.

2.5 Applications

A more recent work [4] at the Oregon Graduate Institute uses ESR techniques in scientific data management. Most of scientific data contain well-defined imprecision either due to the data source (e.g., from the physical world) or modeling limitations. This work applies methods from interval arithmetic to design an interval data model and algebra. Operators on imprecise data give bounds on the result imprecision. ESR algorithms are then used to constrain the imprecision propagation.

Son and Koloumbis [22, 23] have successfully used ESR in their approach to replication and concurrency control for real-time databases. Their token-based algorithm combines real-time constraints for resource scheduling with relaxation similar to divergence control methods based on ESR. Their evaluation of performance of a real-time database shows that the ability to control the trade-off between consistency and concurrency yields potentially large benefits.

Almost all of the existing ESR work assumes only one inconsistency dimension for simplicity. For real-time databases there is a need for controlling both the inconsistency in data values and imprecision in serialization time. The serialization time imprecision is a particularly important issue for multi-version databases. This is the focus of active research.

3 Summary of ESR Work

Papers published on ESR in cronological order:

- (1991 May)[17] Introduction of ESR by Pu and Leff, with the application of ESR to replication.
- (1991 September)[11] Introduction of metric space (a property of database state space) to define the applicability of ESR.
- (December 1991)[19] A survey on Heterogeneous and Autonomous Transaction Processing that includes ESR in a sidebar as an example of autonomous TP.
- (February 1992)[24] A methodology by Wu et al to develop centralized divergence control algorithms and several examples including two-phase locking.

- (February 1992)[18] An informal but detailed description of Autonomous TP using ESR.
- (February 1992)[12] A short position paper on Autonomous TP using ESR.
- (June 1992)[22] Replication control using ESR in real-time database systems by Son and Koloumbis.
- (August 1992 and April 1993)[13] Discussion of the serializability bottleneck in distributed systems and how ESR could alleviate it.
- (April 1993)[23] A specific divergence control algorithm based on tokens for real-time distirbuted databases by Son and Koloumbis.
- (April 1993)[10] Evaluation of divergence control performance with hierarchical inconsistency bounds by Kamath and Ramamritham.
- (May 1993)[15] Distributed divergence control algorithms by composing centralized divergence control, demarcation protocol, and supeordatabase architecture.

Current work:

- Formal characterization of ESR [21].
- Consistency restoration algorithms [9].
- Implementation of centralized divergence control in Encina [8].
- Multiversion divergence control for time fuzziness.
- Performance evaluation of divergence control using simulation.
- Performance evaluation of distributed divergence control (centralized divergence control plus demarcation protocol) using simulation.
- Design of a general ESR system that includes both divergence control and consistency restoration.

References

[1] R. Agrawal, M.J. Carey, and M. Livny. Concurrency control performance modeling: Alternatives and implications. *ACM Transactions on Database Systems*, 12(4):609–654, December 1987.

[2] B.R. Badrinath and K. Ramamrithm. Semantics-based concurrency control: Beyond commutativity. *ACM Transactions on Database Systems*, 16, September 1991.

[3] D. Barbara and H. Garcia-Molina. The demarcation protocol: A technique for maintaining linear arithmetic constraints in distributed database systems. In *Proceedings of the International Conference in Extending Database Technology*, Vienna, March 1991.

[4] R. Barga and C. Pu. Handling inconsistency in scientific data management. Technical Report OGI-CSE-93-005, Department of Computer Science and Engineering, Oregon Graduate Institute, 1993.

[5] P.A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley Publishing Company, first edition, 1987.

[6] Shu-Wie Chen. *The Implementation of A System Supporting Epsilon Serializability.* PhD thesis, Department of Computer Science, Columbia University, Expected, 1993.

[7] S.W. Chen and C. Pu. The implementation of divergence control algorithms in encina. Department of Computer Science, Columbia University; April, 1993.

[8] S.W. Chen and C. Pu. Implmentation of divergence control in encina. Technical Report OGI-CSE-93-XXX, Department of Computer Science and Engineering, Oregon Graduate Institute, 1993.

[9] P. Drew and C. Pu. Asynchronous consistency restoration under epsilon serializability. Technical Report OGI-CSE-93-004, Department of Computer Science and Engineering, Oregon Graduate Institute, 1993. Also available as tech. report HKUST-CS93-002, Department of Computer Science, Hong Kong University of Science and Technology.

[10] M. Kamath and K. Ramamritham. Performance characteristics of epsilon serializability with hierarchical inconsistency bounds. In *Proceedings of the Ninth International Conference on Data Engineering*, Vienna, April 1993.

[11] C. Pu. Generalized transaction processing with epsilon-serializability. In *Proceedings of Fourth International Workshop on High Performance Transaction Systems*, Asilomar, California, September 1991.

[12] C. Pu. Asynchronous transaction execution with epsilon-serializability. In *Proceedings of 1992 Workshop on Heterogeneous Databases and Semantic Interoperability*, Boulder/CO, February 1992.

[13] C. Pu. Relaxing the limitations of serializable transactions in distributed systems. *Operating Systems Review*, 27(2):66–71, April 1993. Also appeared in the Proceedings of Fifth ACM SIGOPS European Workshop, 1992, Le Mont Saint-Michel, France.

[14] C. Pu and S.W. Chen. Implementation of a prototype superdatabase. In *Proceedings of the Workshop on Experimental Distributed Systems*, Huntsville, Alabama, October 1990.

[15] C. Pu, D. Florissi, P. Soares, P. S. Yu, and K.L. Wu. Performance comparison of sender-active and receiver-active mutual data serving. *Concurrency: Practice and Experience*, To appear 1993. Special issue on High Performance Distributed Processing.

[16] C. Pu, W.W. Hseush, G.E. Kaiser, P. S. Yu, and K.L. Wu. Distributed divergence control algorithms for epsilon serializability. In *Proceedings of the Thirteenth International Conference on Distributed Computing Systems*, Pittsburgh, May 1993.

[17] C. Pu and A. Leff. Replica control in distributed systems: An asynchronous approach. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*, pages 377–386, Denver, May 1991.

[18] C. Pu and A. Leff. Autonomous transaction execution with epsilon-serializability. In *Proceedings of 1992 RIDE Workshop on Transaction and Query Processing*, Phoenix, February 1992. IEEE/Computer Society.

[19] C. Pu, A. Leff, and S.W. Chen. Heterogeneous and autonomous transaction processing. *IEEE Computer*, 24(12):64–72, December 1991. Special issue on heterogeneous databases.

[20] Calton Pu. Superdatabases for composition of heterogeneous databases. In Amar Gupta, editor, *Integration of Information Systems: Bridging Heterogeneous Databases*, pages 150–157. IEEE Press, 1989. Also appeared in Proceedings of Fourth International Conference on Data Engineering, 1988, Los Angeles.

[21] K. Ramamrithan and C. Pu. A formal characterization of epsilon serializability. Technical Report CUCS-044-91, Department of Computer Science, Columbia University, 1991.

[22] S.H. Son and S. Koloumbis. Replication control for distributed real-time database systems. In *Proceedings of the Twelfth International Conference on Distributed Computing Systems*, pages 144–151, Yokohama, Japan, June 1992.

[23] S.H. Son and S. Koloumbis. A token-based synchronization scheme using epsilon-serializability and its performance for real-time distributed databases. In *Proceedings of the Third International Symposium on Database Systems for Advanced Applications (DASFAA '93)*, Taejon, Korea, April 1993.

[24] K.L. Wu, P. S. Yu, and C. Pu. Divergence control for epsilon-serializability. In *Proceedings of Eighth International Conference on Data Engineering*, pages 506–515, Phoenix, February 1992. IEEE/Computer Society.