Research Statement

Serafettin Tasci
CSE Dept. of University at Buffalo, SUNY, 338 Davis Hall - Buffalo, NY 14260

Concurrent execution is a big challenge for distributed systems programming and cloud computing. Concurrency needs to be boosted to improve the system performance and availability. For embarrassingly parallel data processing applications, concurrency is boosted and scalability is achieved by adding more workers [1]. However, for data processing applications that require tight synchronization, such as large-scale graph processing systems, real-time enterprise analysis systems, banking and stock market applications, uncontrolled concurrency wreaks safety violations. In these kind of applications, concurrent execution needs to be controlled to prevent latent race conditions and synchronization bugs.

Current Research

My current research focuses on boosting concurrency in distributed systems ranging from large-scale graph processing systems to distributed transactions and wireless sensor networks. In large-scale graph processing my main interest lies in achieving not only serializable but also efficient systems by focusing on improving locality and avoiding nonlocal operations. In distributed transactions, the main purpose is to build a fault-tolerant master-based lock management service which employs intelligent strategies for lock migration and minimal lock durations.

Large-scale Graph Processing: Large-scale graph processing finds several applications in machine-learning, distributed simulations, web-search, and social-network analysis. The significance of these applications led to the development of several graph processing frameworks recently. Most adopt the Bulk Synchronous Parallel (BSP) approach, Pregel framework from Google being a very popular example. However, the max-parallel execution model used in BSP is not suitable for writing applications that require coordination between vertices. For developing applications that require coordination, the serializability semantics (also known as interleaving or sequential consistency) is necessary. In [1], we presented a simple extension to achieve serializability in BSP-based frameworks while keeping the highly-parallel and bulk-efficient nature of BSP executions. We implement our extension on the Apache Giraph framework, an open-source implementation of Pregel, and call the extended framework Giraphx. To provide inter-worker serializability, we augmented Giraphx with two alternative coordination mechanisms: dining philosophers and token ring. In addition, we alleviated the cost of these coordination mechanisms and also improved the performance of Giraph by enabling direct memory reads for intra-worker vertex communication. Our experimental results showed that it is possible to achieve serializability in BSP model while preserving the performance of BSP model.

In our subsequent study [2], we evaluated the suitability of Hazelcast, an in-memory data grid, for large-scale graph processing. We find that, although it is simple to implement a Hazelcast-
based offline distributed graph processing framework, it is hard to achieve comparable performance to the specialized graph processing frameworks (such as Giraph [4]) that are optimized for batch processing of vertices. On the other hand, we find that for a slightly different version of the graph processing problem, i.e., that of online graph processing; Hazelcast excels over offline graph processing frameworks.

**Enabling highly-scalable distributed transactions:** For embarrassingly parallel data processing applications, it is easy to boost concurrency and achieve scalability simply by adding more workers. However, for cloud computing applications that require tight synchronization, achieving concurrency while preserving correctness remains a big challenge. Using locks and transactions are the two most common techniques for developing applications that require tight synchronization. Unfortunately, locking is manual and error-prone. Transactions, on the other hand, are infamous for being unscalable. This project suggests that it is possible achieve scalability for transactions, and aims to design and develop such a framework, called Maestro.

In [3], we proposed the programming model for Maestro. In this initial design, Maestro was planned to use an automated locking service based on predicated locking of program actions achieve a higher level of concurrency. It would analyze the program actions and partition them in terms of their locality and concurrency tolerance and then automatically place locks on the actions in a minimal but safe manner. We implemented the basic concepts of this design to show how this partitioning can be done by demonstration on three case studies. The results revealed that we can skip a significant number of locks in a program thanks to Maestro’s automated locking mechanism.

In the current design, we execute all operations on the shared variables as distributed transactions. Locks are inherently a part of the transactions and every transaction is executed locally. Currently, we are working on the implementation of Maestro leveraging the Hazelcast platform, a popular lightweight open-source in-memory data grid for Java. We are planning to achieve a scalable transactional system by using a lock service architecture with a master and several workers. Keeping the master as the centralized lock master eliminates the complexity and cost of distributed transaction solutions, and enables the master to see all transaction access patterns so that the master can improve access and lock locality of transactions by migrating locks and data when appropriate. At this end, we will use a dynamic lock leasing and data-owner changing scheme that is coordinated by the master.

**Improving concurrency exploiting receiver-side collision detection in wireless sensor networks:** In wireless sensor network (WSN) deployments, Receiver-side Collision Detection (RCD) can be used to speed up collaborative feedback collection from a single-hop neighborhood. Using RCD, an initiator node can query the existence of a predicate P in its neighborhood in constant time by making all P-positive nodes answer simultaneously. Despite the collisions, the initiator is still able to infer useful information from a broadcast using RCD: an activity in the network means the predicate P holds for at least one node while silence indicates that P does not hold at any queried node in the network.
In [4], we investigated the threshold querying problem, where the initiator has to learn whether P holds in the network for at least threshold t number of nodes in single-hop of the initiator. To answer the threshold queries in an efficient fashion, we presented a number of adaptive RCD-based querying algorithms that dynamically re-groups the queried nodes in the network. Experiments on real sensor network deployments revealed that our algorithms achieve significant time improvements in threshold queries over traditional techniques.

**Future Research**

In the rest of my PhD study, I am planning to continue working on the design and implementation of Maestro framework concentrating specifically on the following tasks:

**Improving lock locality:** In Maestro, data is distributed to workers via consistent hashing to keep load-balanced workers. To allow transactions, Maestro maintains a lock for every data item stored in the system. By default, the lock is kept where the data is, and this is very advantageous for improving lock-locality and enabling worker-local transactions without the need for network communication. However, if data items at other workers need to be accessed as part of a transaction, the transaction contacts the master, which serves as the lock owner. If the requested locks are at the master, the master responds affirmatively or negatively depending on the state of the locks. Otherwise, the master requests the locks from the corresponding workers first and starts hosting these locks. As the centralized authority for assigning locks, the master can learn the access patterns of transactions at runtime and manage migration of data/locks to workers to improve access/lock-locality. The master even has the flexibility to move the data so that the data items that are accessed together can be assigned to same workers to increase the access locality.

In this task, we will first identify under what conditions (e.g., how many variables per transaction, contention rate, round trip time between workers, etc.) the centralized lock master solution become more efficient. And then, we will devise lock-caching strategies between the master and the workers based on the characteristics of workload involved in the system. In addition, the master may choose to migrate some locks (and associated data items) from other workers to a transaction-initiator worker. We will identify the most suitable strategies for lock/data migration by using simple regular expression matching strategies (such as 2 consecutive accesses from a worker for the set of data items) as well as more sophisticated strategies that involve performing machine learning over lock access logs at the master.

**Avoiding busy waiting and minimizing lock durations:** Instead of unnecessarily locking variables by some workers periodically/continuously just to evaluate a condition only to find it false and abort the remainder of computation, it is possible to pre-evaluate the conditions required for successful completion of a transaction by read-only access to shared variables.

In addition, we will use callback-based locks, where the master delivers the lock to the requester when it becomes available instead of continuously checking or waiting for a lock that is not currently available.
Finally we will seek for ways to achieve proactive locking in which a lock-service master can anticipate a future lock request and provide the locks ahead of time. By this way, latency can be reduced and throughput can be improved.

**Fault tolerance:** Since transactions are generally used for more sensitive and mission-critical data, it is crucial to ensure Maestro will preserve consistency and deal with failures in the system successfully. This task can be divided to three sub-problems: For master failures, we will adopt efficient extensions of chain replication such as parallel chaining. We expect that the parallel chain replication would benefit over single-chain replication by reducing latency from the sum of latency to the latency of the slowest path and giving us built-in failure detection and simplifying the failure recovery of a failed node.

For worker failures, we need to solve two main issues: releasing locks held by the failed worker, and starting the new worker with updated data from the failed worker. For solving the first issue, we will use heart-beat messages. If a worker fails to communicate with the master for a certain duration despite multiple attempts, the worker releases all the locks it holds. Regarding the second issue, we will explore efficient ways for data replication and logging.

**Past Research**

My work before PhD consists of two main parts: information retrieval and machine translation.

**Information Retrieval:** Information retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources. In my master thesis, I worked on a fundamental information retrieval problem: automatic text categorization which is the task of automatically assigning unlabeled text documents to some predefined category labels by means of an induction algorithm.

In [5] and [6], we made an extensive evaluation and comparison of the popular feature selection metrics in text categorization. In addition to the evaluation of the policies, we proposed new feature selection metrics which show high success rates especially with low number of keywords. Moreover, we proposed an adaptive keyword selection scheme based on selecting different number of terms for each class. Experiments showed that significant improvement can be achieved on skewed datasets which have a limited number of training instances for some of the classes.

In a following study [7], we used the hidden topic analysis of latent dirichlet allocation (LDA) for feature selection and compared it with the popular feature selection metrics in text categorization. LDA is a generative graphical model that can be used to model and discover the underlying topic structures of textual data. We observed that almost in all metrics, information gain performs the best at all keyword numbers while the LDA-based metrics perform similar to chi-square and document frequency thresholding.

**Statistical Machine Translation:** In my work as a researcher in TUBITAK (The Scientific and Technological Research Council of Turkey), I worked on the design and implementation of a
Turkish-English speech translation system. For this project, I implemented a high variety of natural language processing methods such as language modeling, named entity recognition and morphological segmentation and part-of-speech tagging. In addition, I used machine learning methods like expectation maximization and hidden markov models.

In a previous work [8], I worked on the compilation of a bilingual Turkish-English corpus and then implemented a method for sentence alignment based on location and length information in the texts. The content of the corpus was collected from several sources of different genre and required significant text processing for data preprocessing. To the best of our knowledge, this was the first comprehensive bilingual corpus between these languages. In addition, the proposed sentence alignment algorithm was tested on the corpus and success rates up to 96% were obtained.

**Related Publications**


