Scalable Distributed Two-Layer Data Structures (SD2DS)

Krzysztof Sapiecha, Department of Computer Science, Kielce University of Technology, Kielce, Poland
Grzegorz Lukawski, Department of Computer Science, Kielce University of Technology, Kielce, Poland

ABSTRACT

Scalability and fault tolerance are important features of modern applications designed for the distributed, loosely-coupled computer systems. In the paper, two-layer scalable structures for storing data in a distributed RAM of a multicomputer (SD2DS) are introduced. A data unit of SD2DS (a component) is split into a header and a body. The header identifies the body and contains its address in a network. The headers are stored in the first layer of SD2DS, called the component file, while the bodies are stored in the second layer, called the component storage. Both layers are managed independently. Details of the management algorithms are given, along with SD2DS variant suitable for storing plain records of data. The SD2DS is compared to similar distributed structures and frameworks. Comparison considerations together with test results are also given. The results proved superiority of SD2DS over similar structures.

Keywords: Computer Science, Databases, Distributed RAM, Multicomputers, Scalability, Scalable Distributed Data Structures, Two-Layer Design

INTRODUCTION

Scalability, besides high performance and fault tolerance, is an important feature of modern applications (Tanenbaum & van Steen, 2006; Alvin et al., 2010; Lavinia, Dobre, Pop, & Cristea, 2011). Scalable software applications may be distributed over nodes of a multicomputer. Such loosely-coupled architecture is very cost effective and easy to assemble. The only extra effort that should be made consists in developing additional middleware or/and software tools for managing cooperating multicomputer’s resources (Sbalzarini, 2010). Such tools and frameworks may assist developers at different levels of programming, requiring more or less attention to be devoted to the problem of scalability.

At a very low level, simple libraries may be applied, such as MPI (Message Passing Interface) and PVM (Parallel Virtual Machine) (Gropp & Lusk, 1998). They are universal but have to be used by a skilled programmer. They enable the application of message-passing programming model for low-level programming. Both libraries are flexible and universal because MPI and PVM assume that there is no shared memory, and a topology of the network is not important. Actually, the final scalability of an application using PVM or MPI strongly depends
on the skills of the programmer. On the other hand, it is the most flexible approach, and the scalability may concern many different aspects and resources (memory, CPU, disk, etc.).

Google MapReduce (Dean & Ghemawat, 2004) is a framework and a programming model. A problem to be solved is split into sub-problems (in “Map” stage) and processed separately by multicomputer nodes. Then the sub-results are combined (in “Reduce” stage) to obtain the final solution of the problem. Pattern matching and sorting are examples of problems which can be solved in this way. MapReduce is highly scalable and may efficiently utilize a large number of the nodes. Summarizing, it requires less programmer’s efforts to build a scalable application, but it is well-suited to specific problems only.

The above solutions may be helpful in developing a scalable application, where scalability concerns different resources and also CPUs computing power. An important resource of a scalable application may be distributed RAM used as a data storage. It has many advantages over a hard-disk, including very fast access to data and possibility to process the data simultaneously by many nodes of a system running the application. Thus, the usage of a RAM as data storage, instead of a hard-disk is investigated in the paper.

Distributed Shared Memory system (DSM) (Nitzberg & Lo, 1991), like IVY (Li & Hudak, 1989), simplifies the process of developing an application. It creates a shared memory out of separated RAMs of multicomputer nodes. Moreover, applications designed for typical shared memory systems (even home computers) may be easily adopted for a multicomputer by means of DSM. The DSM may be implemented in hardware and/or in software. The DSM may be used to develop a scalable application. On the other hand, its usage is limited, and may become a bottleneck if inefficiently used.

Scalable Distributed Data Structures (SDDS) (Litwin, Neimat, & Schneider, 1996) are a middleware that can use distributed RAM as a scalable file of records. An SDDS file may be implemented in any programming language, may be used with many applications, even as a block device for a filesystem (Chrobot, Lukawski, & Sapiecha, 2008) or as a part of a database (for example, AMOS (Ndiaye, Diene, Litwin, & Risch, 2001)). However, all SDDS architectures require a large number of records to be moved between buckets (located on nodes of a multicomputer) during data expansion (Litwin, Neimat, & Schneider, 1996). Moreover, the addressing rules should take into account characteristics of the data, e.g. a probability distribution of record keys. Otherwise, the RAM may be used inefficiently. These are serious drawbacks that considerably diminish the area of SDDS applications, despite their fast access to the buckets.

Scalable Distributed Two-layer Data Structures (SD2DS) (Sapiecha, 2010) which overcome the shortcomings are introduced in the paper. Just like SDDS, they are highly scalable and accessed fast. However, only very small amount of data is moved between buckets during the expansion. Hence, the RAM is used efficiently, even for the most inconvenient probability distribution of the keys. Moreover, SD2DS may apply different memory management policies, so it may fit different applications. An access to data stored in SD2DS uses indirect addressing. It may be compared to indexing used in databases. However, it is operated differently and may also be used for load balancing and fault tolerance of a system bearing SD2DS. The overhead of indirect addressing is negligible, if the advantages of SD2DS are taken into account as shown in the paper.

The first section contains brief description of SDDS. Motivation for the research is given afterwards. Followed by a section which SD2DS are defined. SD2DS for records are compared with SDDS in the next section. The paper ends with conclusions.

RELATED WORK

The basic SDDS element is a record, and each record is uniquely identified with a key. Records with keys are stored in buckets. The buckets
Related Content

Service and Management Oriented Traffic Information Grid
[www.igi-global.com/article/service-management-oriented-traffic-information/47424?camid=4v1a](http://www.igi-global.com/article/service-management-oriented-traffic-information/47424?camid=4v1a)

Optimizing Communication for Multi-Join Query Processing in Cloud Data Warehouses

Consistency of Replicated Datasets in Grid Computing
[www.igi-global.com/chapter/consistency-replicated-datasets-grid-computing/20508?camid=4v1a](http://www.igi-global.com/chapter/consistency-replicated-datasets-grid-computing/20508?camid=4v1a)
Multiuser Diversity OFDMA using Power Priority Selection and Adaptive Clipping
International Journal of Distributed Systems and Technologies (pp. 18-30).
www.igi-global.com/article/multiuser-diversity-ofdma-using-power-priority-selection-and-adaptive-clipping/119191?camid=4v1a