High-Speed Information Utilization Technologies Adopted by Interstage Shunsaku Data Manager

● Hiroya Hayashi

(Manuscript received November 28, 2003)

The current business environment is characterized by many difficulties, including ongoing economic stagnation, fierce competition, and rapid staff turnover. The key to business survival is to gather as much information as possible, to utilize it as effectively as possible, and to be able to tie this information to business operations. However, at present the data integration and management technologies for utilizing a diverse range of business information are still incomplete. While attempting to find a solution to this problem, our attention was drawn to XML, and we developed Interstage Shunsaku Data Manager-an XML database engine that achieves high-speed searches of XML data. This paper explains the features of Interstage Shunsaku Data Manager and the high-speed information utilization technologies that it employs.

1. Introduction

The key to business survival in the current environment is to gather as much information as possible and utilize it as effectively as possible. To achieve these aims, particular attention is being paid to how information utilization can be directly linked to business activities in the field. However, data integration and management technologies suitable for a diverse range of business information have yet to be perfected.

Until now, Relational Database Management Systems (RDBMSs) have generally been adopted as solutions for data integration and management. However, RDBMSs require various schemas and data structures for integrating business data (data normalization), which makes designing such systems difficult. Also, performance tuning such as adding indexes to the search target data is required to search the data quickly. Indexes can be effective for performing high-speed searches on particular data items, but these techniques cannot be used on data items that do not have indexes. Another problem is that if data items are changed, then the indexes also need to be redesigned. These issues present difficulties when it comes to designing and operating RDBMSs.

Our attention was drawn to XML because of its potential in terms of data integration. XML prescribes simple data structures whereby data items are enclosed by start and end tags. This eliminates the need to normalize data and design integrated schemas, even when integrating different types of data. All that is required is to add tags to the data and to group the data into a single document. XML can also use recursive structures to deal flexibly with variable-length and variable-item data, which are not handled well by RDBMSs. However, despite XML's flexibility, the XML data management systems that have been developed up to now have had unresolved issues regarding search performance.

We developed Interstage Shunsaku Data Manager (hereafter referred to as Shunsaku) as an XML database engine that can search data

quickly while still retaining the flexibility of XML.[1)-3)] This was achieved using a combination of our original SIGMA data search algorithm and parallel search technology using blade servers.

Shunsaku combines XML's flexibility with high-speed data search technology, both of which are important for data integration and management and suitable for using diverse business information in the field. Shunsaku can also further expand the scope of XML utilization.

The following sections outline Shunsaku's architecture and its operation features.

2. Shunsaku's architecture

This section describes the architecture of Shunsaku.

Shunsaku systems are made up of director servers and search servers. Shunsaku employs an architecture whereby search servers, each equipped with a search engine, are arranged in parallel and can execute parallel searches. It is assumed that these parallel search servers will be blade servers.

The role of each type of server is explained below. **Figure 1** shows the architecture used by Shunsaku.

1) Director servers

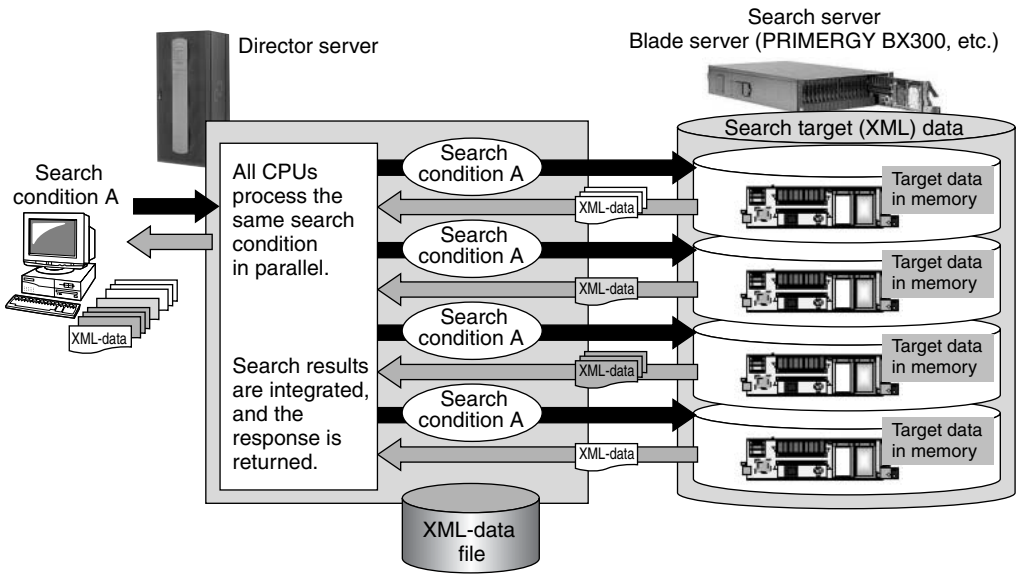
Director servers have functions for coordinating multiple search servers in parallel and do the following:

- Manage files that store the search target data
- Distribute the search target XML data evenly to each search server
- Receive search requests from applications
- Distribute search requests to the search servers
- Merge search results from the search servers and return the merged result to the application that requested the search

2) Search servers

Search servers have search processing functions for the target XML data and do the following:

- Search XML data at high speed by using the SIGMA search algorithm.
- Receive and store target XML data from the director server. As a result, there are no disk I/O operations during the search stage.
- Receive search requests from the director



Director server: Receives search requests and returns search results.
Search server: SIGMA high-speed, XML-data search engine.

Figure 1
Architecture of Interstage Shunsaku Data Manager.

server and return search results to the director server.
- Enable advanced parallel searches. This is achieved by having each search server run independently.
- Enable several hundreds of search requests to be processed in parallel. This is achieved by using low-cost blade search servers.

3. Shunsaku's operation features

Shunsaku can treat XML data as text. This allows the flexibility of XML to be exploited and, in terms of data integration, eliminates the need to design data normalization and integration schemas for various types of data. Also, because XML can easily handle variable-length and variable-item data (unlike RDBMSs), it is not limited to data integration because it can be expanded into other applications. In addition, Fujitsu's original SIGMA algorithm makes indexless searches possible, which means that every item (character, numeric, date, etc.) of XML data can be searched. Furthermore, by combining parallel search architecture with blade servers, both the performance and size of Shunsaku systems can be scaled up easily.

These features mean that the construction and operating costs of Shunsaku systems can be from 30 to 70% lower than those of a conventional RDBMS alternative.

The following sections give further details of Shunsaku's features.

3.1 Easy changes to data items

Shunsaku can perform indexless searches on XML data by using search engines equipped with the SIGMA algorithm.[4] This means that all XML data items can be searched. If search data is added or changed either during or after system construction, the system can respond flexibly to such additions and changes without the need to redesign the database.

Next, we describe the SIGMA algorithm, which is Shunsaku's core technology.

The SIGMA algorithm is a high-speed string lookup algorithm that uses unidirectional sequential processing. This algorithm is installed on the search servers and used to search XML data (**Figure 2**).

The SIGMA algorithm performs the follow-

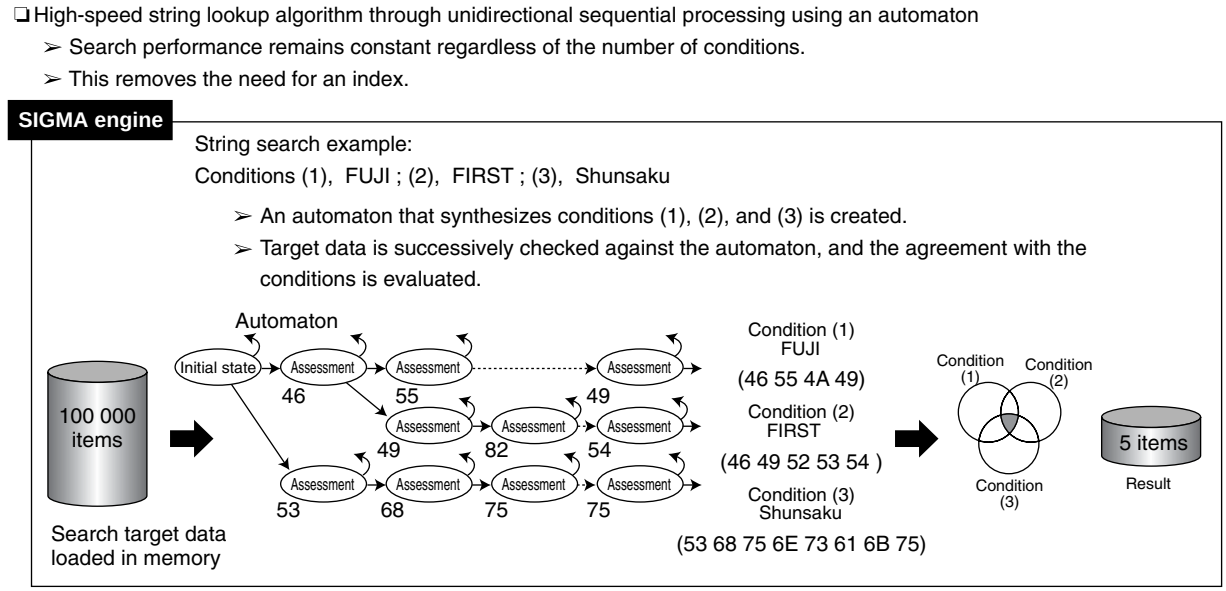


Figure 2
SIGMA algorithm.

ing steps:

1) Disassembles the search conditions in byte units and creates an automaton for the search conditions.
   In the example shown in Figure 2, condition 1, "FUJI", is evaluated as "46 55 4A 49"; condition 2, "FIRST", is evaluated as "46 49 52 53 54"; and condition 3, "Shunsaku", is evaluated as "53 68 75 6E 73 61 6B 75". Then, an automaton is created as shown in the figure.
2) Evaluates match conditions by successively comparing each byte of the data in memory (from beginning to end) against the automaton created in Step (1).
   In the example, if the value of the byte being analyzed is "46", the automaton switches to the next state. If the next value is "55", the automaton again switches to the next state. This process continues until all of the data has been evaluated. If a value does not meet the automaton state transition conditions, the automaton returns to its initial state. When a block of data has been completed, the result state of the automaton is evaluated and the algorithm determines whether the data is a hit (whether it matches the search conditions).

With the SIGMA algorithm, the search is completed as soon as it has run through the search target data once. The search performance of this algorithm is not affected by the number of hits, because it always runs through all of the data once, regardless of the number of hits. Furthermore, consistent search performance can be maintained no matter how many search conditions there are, because only the width of each automaton increases with the number of search conditions and there is no change in the cost of evaluating data. The strengths of the SIGMA algorithm are that any data item can be a search target and indexes are not required for searching, because every search always runs through all of the data.

Shunsaku loads search target data into the memory of each search server and then executes the SIGMA algorithm. This means that search performance depends purely on the amount of data loaded in memory and the performance of the CPU that is processing it. That is, the search performance increases as the CPU performance increases–if the CPU performance is doubled, the search performance will also double.

3.2 Superior multiplexing performance

In the implementation of the SIGMA algorithm used by Shunsaku, all of the search target data is loaded into memory, so the CPU usage rate for search processing is always 100%. As a result, new search requests cannot be received while another search request is being processed. This means that if each search process takes an average of one second to complete and 100 search requests are issued simultaneously, then the 100th request will take 101 seconds to process (**Figure 3**).

In such situations, Shunsaku's original high-traffic technology delivers superior multiplexing performance. Figure 3 shows this technology, which is explained below.

The search performance of the SIGMA algorithm does not depend on the number of search conditions. This characteristic can be exploited to chain together multiple simultaneous search requests using "OR" conditions. In this way, a single search condition can be formed so that all of the search requests can be processed simultaneously. The time taken for this kind of batch processing does not differ from that for a single search request, thanks to the characteristics of the SIGMA algorithm. When the search process is completed, the results are distributed for each request.

Requests that are issued when search processing is already underway are kept on hold until the processing for the preceding search request is completed. If multiple search requests are issued during this time, then all of these requests are grouped together and processed as a batch.

The amount of time that searches can be ex-

pected to take using high-traffic technology is the time taken to keep the request on hold plus the time taken to process the search itself. If, for example, it takes an average of one second to process a search, then the search can be expected to take two seconds (one second to keep the request on hold plus another second to process the search itself). **Figure 4** compares the performance obtained using multiplexing and a conventional RDBMS. For example, if there are 100 concurrent requests, the processing time (6.63 s) is only about 4.5 times the time (1.46 s) required for a single request, which is a remarkable improvement over conventional RDBMSs.

3.3 Simple scale-up operations

Shunsaku achieves parallel search processing by utilizing blade servers for the search servers that run the search engines. The search processing performance solely depends on the amount of search target data loaded into memory and the performance of the CPUs. For example, suppose that there is 50 MB of search target data, that this data is loaded into a single search server, and that it takes one second to process a search. Then, if the amount of data increases to 100 MB after system operations commence, the time taken for each search simply increases to two seconds. At this point, if another search server (blade server) is added, then each search server (blade server) will hold 50 MB of data, the time taken to process each search will be back to one second, and the search performance is maintained. In this way, Shunsaku systems can be scaled up simply by adding search servers (blades) as the amount of data increases.

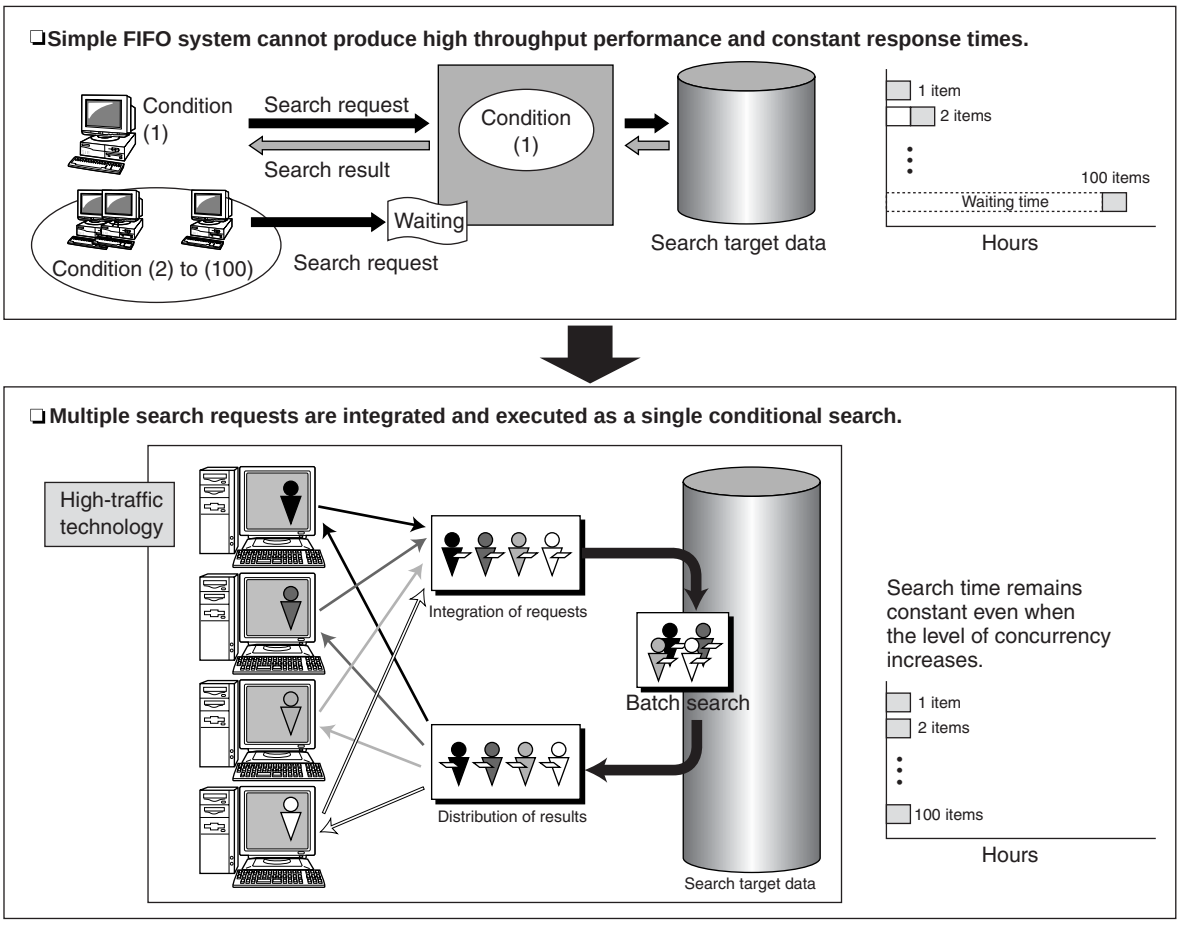
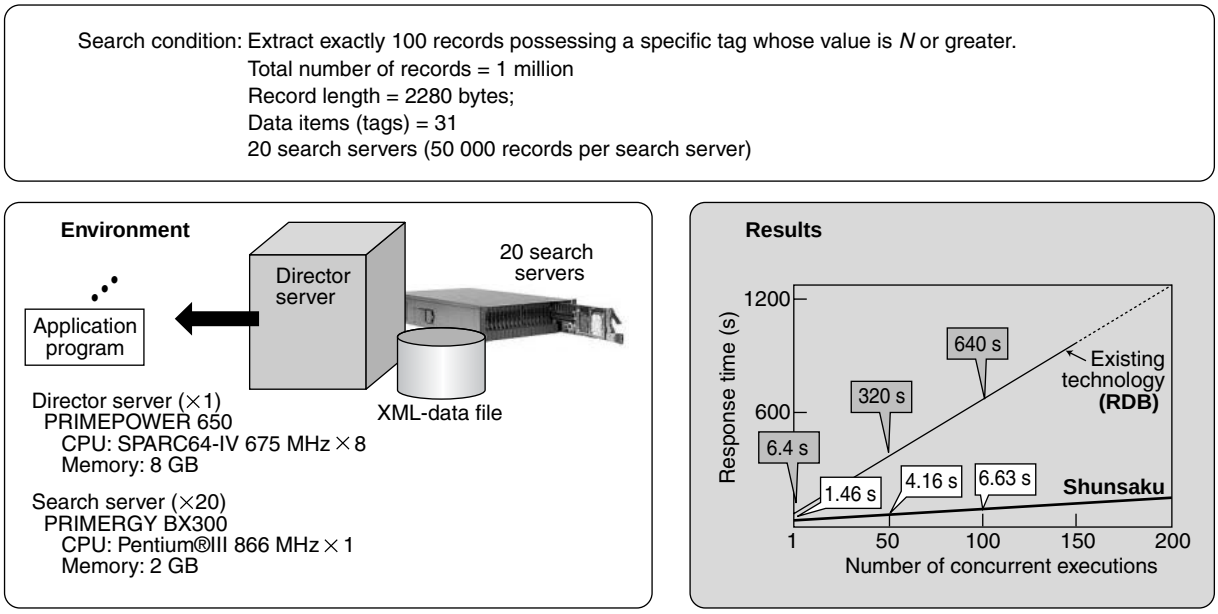


Figure 3
High-traffic technology.



Note: In the measurement environment, the number of records that matched the search condition was set at 1000.

Figure 4
Test results (benefits of using high-traffic technology).

In addition, if a Shunsaku system is operating with multiple search servers and one of the search servers fails, the director server automatically detects the failure and redistributes the data held by that search server among the other search servers (automatic degradation). As a result, if a search server fails, processing can be continued without needing to stop the system.

4. Future development scenarios

Shunsaku provides basic functions that focus on search processing. Search data is updated and added using command-based batch processing. In the future, users will become more concerned about whether search data is up to date. To ensure that search data is up to date, the next step is for Shunsaku to achieve real-time data updates. The basic requirement for achieving this goal is to ensure that the files Shunsaku uses to hold the search target data are robust. To achieve this, we will introduce the concept of transactions for data updates. Achieving real-time updates is likely to further expand the range of Shunsaku applications.

The current version of Shunsaku assumes that data is concentrated at the location where the Shunsaku system is installed. However, the reality is that the cost of the operations for collecting data from each site is often high. In response, we plan to implement a "data grid" concept in Shunsaku so that data distributed over multiple sites can be easily searched from any site.

5. Conclusion

We have developed an XML database engine, called Interstage Shunsaku Data Manager, to overcome the shortcomings of the data integration and management technologies that are suitable for using diversified business information. Shunsaku enables high-speed data searches while making the most of XML's flexibility. It has a parallel search engine equipped with Fujitsu's original SIGMA algorithm. These features enable Shunsaku to handle variable-length and variable-item XML data without modifications and perform high-speed searches of all data items. Shunsaku is well suited to production systems, because it can easily be scaled up in response to increases in

the amount of data and provides functions such as automatic degradation that ensure availability. By using Shunsaku as the core engine, optimal data integration/management solutions can be created, while further expanding the use of XML.

In the future, Shunsaku will be further developed to include support for real-time updates and will evolve to include the data grid concept, which will further expand the scope of Shunsaku applications.

Hiroya Hayashi received the M.S. degree in Computer Science from the Graduate School of Engineering at Gunma University, Gunma, Japan in 1987. He joined Fujitsu Ltd., Numazu, Japan in 1987, where he worked on development of an operating system for fault-tolerant systems. He then worked on development of the relational database for the open systems Syfmoware, and is currently working on development of an XML database engine.

References

1) Interstage XML Search (Product: Shunsaku Data Manager V6). 2003. *http://www.fsw.fujitsu.com/products/InterstageSuite/XMLSearch/overview.html*
2) Interstage Shunsaku. (in Japanese), *Fujitsu Journal*, **29**, 5, 2003. *http://journal.fujitsu.com/261/topstory/02.html*
3) Interstage Shunsaku Data Manager. (in Japanese), 2003. *http://interstage.fujitsu.com/jp/v6/shunsaku/*
4) Full-text retrieval. *Fujitsu Journal*, **28**, 1, 2002. *http://journal.fujitsu.com/253e/sp3.html*