https://doi.org/10.20965/jdr.2018.p0338

Paper:

An Analysis Technique of Evacuation Simulation Using an Array DBMS

Yusuke Kawai*, Jing Zhao, Kento Sugiura**, Yoshiharu Ishikawa*,†, and Yukiko Wakita***

*Graduate School of Informatics, Nagoya University
Furo-cho, Chikusa-ku, 464-8601 Japan
†Corresponding author, E-mail: ishikawa@i.nagoya-u.ac.jp
**Graduate School of Information Science, Nagoya University
[Received November 2, 2017; accepted March 2, 2018]

Today, large-scale simulations are thriving because of the increase of computating performance and storage capacity. Understanding the results of these simulations is not easy, and hence, support for interactive and exploratory analysis is becoming more important. This study focuses on spatio-temporal simulations and attempts to develop an analysis technology to support them. It uses a database system for supporting interactive analysis of large-scale data.
Since the data gained via spatio-temporal simulations is not suitable for management in a relational DBMS (RDBMS), this study uses an *array DBMS*, a type of DBMS that has been garnering increased attention in recent years. An array DBMS is designed for the management of large-scale array data; it provides a logical model for array data, yet it also supports efficient query processing. *SciDB* is used as our specific array DBMS in this paper.
This study targets disaster evacuation simulation data and demonstrates via experimentation that the query-processing functions offered by an array DBMS provide effective analysis support.

Keywords: spatio-temporal simulation, array DBMS, evacuation simulations

1. Introduction

Today, large-scale simulations are thriving due to the increase of computating performance and storage capacity. These advancements result in the production of a huge amount of data. When simulations grow to a massive scale, it becomes difficult to understand the results of their analyses, and structures that support those analyses become more important [3, 7, 8]. In particular, to provide flexible analysis, support for the process of analyzing data in an interactive and exploratory manner is required.

This study focuses on large-scale simulations based particularly on spatio-temporal data (*spatio-temporal simulations*), and an attempt is made to develop effective management and analysis technology to support these systems. Numerous scientific simulations involving such areas as weather, disasters, and traffic are conducted, and the obtained data is mapped using temporal and spatial coordinates, enabling the data to be displayed as a large-scale array based on the dimensions of time and space. However, it is difficult to handle this type of data using a traditional relational DBMS (RDBMS) designed to handle table data. Therefore, this study uses the recently introduced array DBMS. An *array DBMS* is specialized to the use data stored in multidimensional arrays. By linking spatio-temporal data to dimensions in multidimensional arrays, analysis based on spatio-temporal data can be efficiently performed. In this paper, we use an array DBMS *SciDB* [1, 2, 5, 6], a large-scale parallel distributed DBMS designed for scientific data analysis. Since SciDB uses data storage methods and query-processing methods tailored to large-scale arrays, it can handle a wide variety of queries for large-scale arrays.

The spatio-temporal data focused in this research is based on an evacuation simulation model for an earthquake in the Tokyo metropolitan area. It tracks the positions and status of individuals 24 hours from the time of the earthquake [4]. We present an analysis example that uses the multidimensional array structure for the simulation data using time and space as dimensions, and consider required operations. We measure the query time for these operations in SciDB and evaluate whether spatio-temporal simulation data can be analyzed efficiently. Based on these results, we discuss the prospects for large-scale spatio-temporal simulation data analysis.

This paper is organized as follows. In Section 2, the evacuation simulation data used in this research is described. Next, in Section 3, we explain the features of SciDB and its basic operations. In Section 4, we describe the analysis model and the system architecture used in this analysis, and the evacuation data analysis example. In Section 5, we present measured query times in SciDB based on the analysis example in Section 4 and evaluate the use of SciDB. Finally, in Section 6, we summarize this paper and describe future challenges.

2. Spatio-Temporal Simulation Data

We explain the spatio-temporal simulation data used in this study. First, we introduce an overview of the model, and then we describe the format of the data.

The simulation data was provided by Osaragi Labo-

Table 1. Simulation data attributes.

Attribute	Type (units)	Explanation
Time step	Real (min.)	In 0.1 min steps for 24 h
ID	Integer	From 1 to 13487
x coord.	Real (min.)	Positive is east
y coord.	Real (min.)	Positive is north
Status	Text	14 types. See **Table 2**.

Table 2. Individual status.

Status	Meaning
StayHome	Staying at home
StayOther	Staying in building other than home
Move	On foot outside during disaster
StayIttoki	Currently staying at rendezvous point
Safe	Staying at wide-area shelter
Search	Searching for evac. route to destination
HinanIttoki	Currently en route toward shelter
HinanIttoki2	Currently arrived at shelter
HinanKoiki	Currently en route toward wide-area shelter
HinanKoiki2	Currently arrived at shelter
Rescue	Participating in rescue operations
Fire	Participating in firefighting operations
Konnan	Trapped on street
Death	Deceased

ratory at Tokyo Institute of Technology and generated based on an evacuation simulation for an earthquake in the Tokyo metropolitan area [4]. The target region is Kita-Senju in Adachi Ward. This region is bounded by the Ara and Sumida Rivers, making wide-scale evacuation to locations outside the area difficult. Furthermore, the wooden structures in the region are closely packed, making it an area at high risk of building collapses and fires during an earthquake. This simulation is based on the hypothesis that an earthquake of a seismic intensity of upper 6 occurred in the northern part of Tokyo Bay at 18:00, and then it considers the condition that fires occurr in the residential housing of the Kita-Senju's central area.

The simulation results are recorded in chronological order in a CSV file. Evaculation records are produced at the interval of 0.1 min during for 24 h, and the status of individuals, buildings, and streets is recorded. In this paper, we only use individuals' evacuation status. The detail of the data is explained below.

The data on individual status contain spatio-temporal information as well as evacuation conditions and other information. The attributes of the data used in the study are listed in **Table 1**. The simulation was conducted with a population of 13,487 individuals, each with a unique ID. Its spatial information is recorded using Plane Rectangular Coodinate System IX. The status of individuals was represented by 14 different types of text strings, such as "staying in building" and "moving to evacuate," as shown in **Table 2**. Even if an individual died during evacuation or evacuated to outside the Kita-Senju area, that individual's actions continued to be monitored over the 24-h period. In brief, the data for 13,487 individuals was recorded over a 24-h period, creating 190 million records.

```
array_name
<attribute1_name : attribute_type, attribute2_name ... >
[ dimension1_name = low:high,chunk_length,chunk_overlap,
  dimension2_name ... ]

attribute_type = (u)int[8-64], float, double, char, string, bool
```

Fig. 1. Array definition syntax in SciDB.

```
CREATE ARRAY array4x4 <val:int64> [i=0:3,4,0, j=0:3,4,0]
store(build (array4x4, i*4+j+1), array4x4)
```

Fig. 2. Example of array creation.

3. Overview of SciDB

This section describes the data format, features, and queries of SciDB. SciDB manages data in a multidimensional array format. Although SciDB has a variety of features, we focus our discussion on those that are relevant to this work.

3.1. Multidimensional Arrays

The array definition syntax in SciDB is shown in **Fig. 1**. An array has dimensions and their corresponding elements. Each element has one or more attributes, and each attribute has a specified type. A dimension value must be an integer, and an element is uniquely determined by specifying the dimensions. Array dimension data in SciDB contains the following information:

- Range of dimension (min. & max. values)
- Chunk length
- Chunk overlap

Chunks are explained in the next subsection.

For example, **Fig. 1** shows an example of array creation, and **Fig. 2** is the corresponding array. **Fig. 1** defines an array consists of two dimensions *i* and *j* with the name `array4x4`. Each dimension has values from 0 to 3, and each element in the array holds an integer value with the name *val*.

3.2. Chunks and Parallel Distributed Processing

One of the major features of SciDB is its partitioning of large-scale array data into fixed-length chunks and their distributed processing. A *chunk* is a unit of physical data management. As elements that are closely positioned in an array are stored in close positions on the disk, efficient processing is possible using a limited part of the data. As for partitioning, the chunk size and overlapping with other chunks can be controlled by the user level. Furthermore, as the overall framework is based on parallel decentralized processing, analytical processing of large-scale data is also supported. Note that there is no specific upper limit on the database storage capacity in SciDB.

3.3. Primary Operations

Since SciDB is a DBMS designed for scientific data analysis, it includes a variety of operations for scientific

i \ j	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

Fig. 3. Two-dimensional array `array4x4`.

i \ j	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	**9**	**10**	**11**	**12**
3	13	14	15	16

(a) slice operation

i \ j	0	**1**	**2**	3
0	1	2	3	4
1	5	**6**	**7**	8
2	9	**10**	**11**	12
3	13	14	15	16

(b) between operation

Fig. 4. Extraction operations.

analysis. It implements everything from simple operations, such as the extraction based on specified conditions, aggregation by dimensions, and insertion/deletion and combination of array elements, to operations requiring complex processing, such as matrix computation and generation of probability distributions. Since the analysis in this study focuses on the extraction and collection of data in particular spatio-temporal ranges, we provide simple introductions to the operations used in SciDB, with a focus on those that appear in this study.

3.3.1. The Slice Operation

The slice operation extracts elements with the specified value in each dimension. For example, the slice operation for extracting elements with the value of 2 in dimension *i* in the array in **Fig. 3** is `slice(array4x4, i, 2)`. The extraction example is shown in **Fig. 4(a)**. Through this operation, the elements $\{9, 10, 11, 12\}$ that have the value 2 in dimension *i* are extracted. Although only one dimension is used in this example, it is also possible to specify values in multiple dimensions. In the resulting array, the dimension specified in the original array is eliminated. In this example, as the dimension *i* was specified, the resulting array of $\{9, 10, 11, 12\}$ is a one-dimensional array consisting only of the dimension *j*.

3.3.2. The Between Operation

The between operation extracts elements in a specified range in each dimension. We specify the upper and lower limits for each dimensional range. For example, the between operation for extracting the elements in the array in **Fig. 3** whose values in dimensions *i* and *j* fall between a lower limit of 1 and an upper limit of 2 is `between(array4x4, 1, 1, 2, 2)`. An extraction example is shown in **Fig. 4(b)**. Through this operation, the elements $\{6, 7, 10, 11\}$, whose values in dimensions *i* and *j* lie in the range where the lower limit is 1 and the upper limit is 2, are extracted.

i \ j	0	1	2	3
0	3.5		5.5	
1				
2	11.5		13.5	
3				

Fig. 5. Regrid operation.

3.3.3. The Regrid Operation

SciDB supports many aggregation operations. These operations calculate aggregated values for the specified dimensional value or range to create new arrays.

The regrid operation aggregates elements in the specified range in each dimension and applies an aggregation function. For example, the regrid operation for identifying and specifying a range of 2 for both dimensions *i* and *j* in the array in **Fig. 3** and applying the avg operation is `(array4x4, 2, 2, avg(val))`. The result of this aggregation operation is shown in **Fig. 5**. This operation requires that a range in each dimension be specified; dimensions where aggregation is not performed should have a specified range of 1.

As another example of an aggregation operation, the aggregate operation aggregates elements in each dimension specified and applies an aggregation function. For example, the aggregate operation for applying a sum function to the elements in each dimension *j* in the array in **Fig. 3** is `aggregate(array4x4, sum(val), j)`.

4. Analysis Example

Here, we describe an analysis example using SciDB featuring spatio-temporal dimensions for the evacuation simulation data in Section 2. First, we explain the spatio-temporal simulation data analysis model of SciDB. Then, we present an example for the evacuation simulation data and perform queries using SciDB.

4.1. Analysis Model

In this research, spatio-temporal simulation data is managed in SciDB using multidimensional arrays. **Fig. 6** presents the image of a multidimensional array for spatio-temporal data in SciDB. Such an array has space and time as its basic dimensions, but we can add other dimensions as needed. The grid cells in **Fig. 6** correspond to the elements of the array and store information necessary for analysis in the elements of the array.

To analyze the simulation data in SciDB, as preprocessing, it is necessary to convert them to an array format and store it in SciDB. As stated in Section 2, the target spatio-temporal information used in the simulation data consists of real numbers, with individuals scattered throughout a continuous space. Therefore, in preprocessing, the simulation data is partitioned into grids, aggregated, and stored in SciDB.

Table 3. Loading time and array size.

	Population						
	2,000	4,000	6,000	8,000	10,000	12,000	13,487
No. of file records in SciDB (×10,000)	2.77	4.61	5.83	7.26	8.14	8.98	9.90
File size (GB)	1.15	2.30	3.45	4.60	5.78	6.93	7.79
Load time (s)	84.4	167.9	245.5	331.9	405.4	482.3	559.1
SciDB array size (MB)	9.8	16.9	21.8	27.4	31.1	35.0	38.7

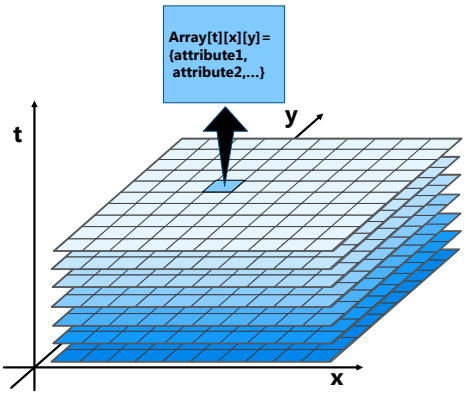


Fig. 6. Image of multidimensional array corresponding to spatio-temporal data.

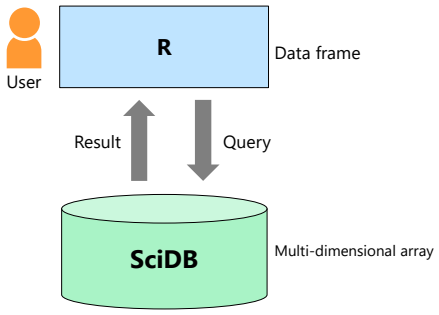


Fig. 7. Analysis system architecture.

Table 3 shows the time required to load the spatio-temporal simulation data described in Section 2 in SciDB. To view the growth of the time required, the number of individuals (number of separate IDs) included in the simulation data is increased from 2,000. For example, the user population = 2,000 means that the user IDs up to 2,000 were loaded. From the table, it is clear that load time[1] increases roughly in proportion to the increase in the number of individuals and file size. **Table 3** also shows the size of each file after loading in SciDB. Since we reduce the amount of data in the preprocessing and SciDB represents data in a binary format and further compress the data, the array size becomes quite compact.

Figure 7 presents the system architecture for analyzing the spatio-temporal simulation data. In this analysis, we use the R language as the interface for SciDB. SciDB is equipped with for R[2]. Array data in SciDB corresponds to tabular data in R called "data frame," and query results obtained from SciDB are processed in R, making data visualization possible.

4.2. Analysis of Evacuation Simulation Data

We analyze the evacuation simulation data described in Section 2. First, an example for the evacuation simulation analysis is provided. Then we consider the granularity of the grids as the analysis is conducted on spatio-temporal grids. Then we shown an example of data analysis.

1. The load time is the sum of the times for loading in R, for preprocessing using R, and for storing in SciDB.
2. https://github.com/Paradigm4/SciDBR

4.2.1. Analysis Target

There are numerous considerations to be made in analyzing evacuation data. Here we consider the total number of individuals based on evacuation conditions and their movement status.

First, we consider counting of individuals by evacuation status. As shown in **Table 2**, the target evacuation simulation model expresses the status of individuals by character strings. They are converted into integer values and treated as dimensions; combined with spatio-temporal dimensions, it creates a four-dimensional array. Since the array elements store the corresponding time steps, spatial grids, and number of individuals in each condition, it is possible to understand how many individuals are in each status by designating a dimension that corresponds to a given status. Example queries are "At what time step will evacuation be x% complete?" and "In which regions will the number of casualties and those who have difficulty evacuating be high?" In addition, to view changes in the number of individuals, we can consider an analysis that takes into consideration the incremental difference between two time steps. Since the sections where the number of individuals changes correspond to points where people are moving or conditions are changing, they are important as evacuation information.

Next, we consider how to capture movement status. We perform preprocessing for the analysis: the distance each individual in the grid has moved after a specified period of time is entered into each element of the array. By observing the sizes of the movement distances, it is possible to understand the trends such as evacuation is being hurriedly conducted and individuals tend to be seeking shelters. Note that these analysis items are not for understanding by viewing numerical values, but through consulting the visualizations discussed below.

Table 4. Element attributes.

Attribute	Type	Description
Population	Integer	No. of people in the cell
x_move	Real	Movement dir. in x coord.
y_move	Real	Movement dir. in y coord.

Table 5. Dimensions.

Dimension	Range	Description
t	[0 : 1440]	Time step (min)
x	[0 : 879]	x coordinate
y	[0 : 443]	y coordinate
status	[1 : 14]	Status (corresp. to **Table 2**)

4.2.2. Grid Granularity

In exploratory analysis of spatio-temporal simulation data, as the desired level of detail for information differs based on analysis conditions, we may change the granularity of the spatio-temporal grid each time. Moreover, when performing exploratory analysis of large-scale simulation data, we often conduct simple analyses first, then perform more-detailed analyses of areas that require further investigation.

SciDB has the regrid operation as an aggregate function; if it takes the sumation fuction sum, it is possible to integrate multiple grids. Once summation is performed and the grids are integrated, the finer information in the original grids is lost. Therefore, we store the smallest grid data used in the analysis in SciDB. In other words, by extracting only the information required at the time of analysis and using regrid to integrate the grids, it is possible to generate grids of the desired granularity dynamically.

4.2.3. Analysis Example in SciDB

We conduct an analysis in SciDB using the evacuation simulation data. First, the array format stored in SciDB is as follows:

```
array <count:int64, x_move:double, y_move:double>
  [t=0:1440,41,0, x=0:879,41,0, y=0:443,41,0,
   status=1:14,14,0]
```

The grid size is 5 m $\times$ 5 m and this is used as the minimum granularity. The elements used in the analysis store the numbers of people in the grid cells and information related to their movements after 1 min as movement states. The sum of the distances individuals in the cell have moved is used as a value expressing movement status; in practice, average movement distance divided by the number of individuals in the cell is used in the analysis. The attributes of the elements in the array are shown in **Table 4** and the dimensions are shown in **Table 5**.

Visualization is an important method for the analysis of large-scale simulation data. Even if the original data is not shown, we can understand the overall trend; it makes visualization an extremely effective approach in data analysis. Visualization is performed using other tools after the required data is extracted from SciDB. In other words, what is demanded for SciDB is quick extraction of the data to be visualized. In this analysis, visualization is performed using plot functions in the R language.

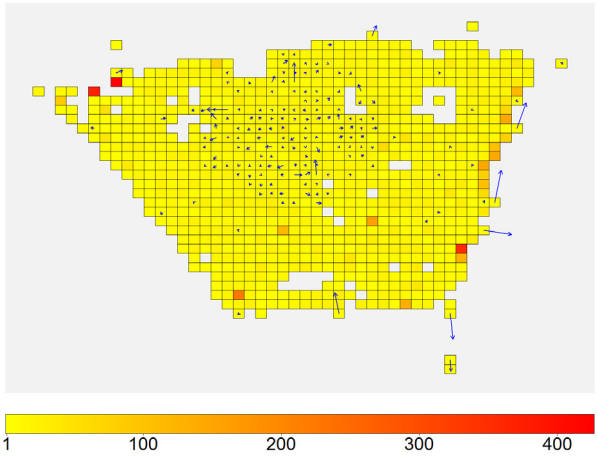
A visualization example of the query result for the analysis described in Subsection 4.2.1 is shown in **Fig. 8**. The time step in **Fig. 8(a)** is $t = 5$ (min); in **Fig. 8(b)**, it is $t = 60$ (min). Both figures present the number of individuals and movement conditions and they are visualized on a 25 m $\times$ 25 m grid. The arrows indicate the state of movement after 1 min, and the individuals in the starting grid cell are represented as moving toward the endpoints. The lengths of the arrows correspond to the average movement distance for each minute. The grid colors indicate the number of people, as shown in the color bars in the figures. The maximum number of people in the time step is indicated in red; the minimum number of people is indicated in yellow.

We mention what can be understood when the spatial grid is visualized via time steps. If the arrows continue to form one big arrow, this line indicates the route used in evacuations, and the end points of large arrows would be evacuation points. In addition, we can determine whether evacuation is being actively performed at that particular time step by observing the number of arrows. To cite a specific example, in **Fig. 8(a)**, many arrows are visible in the top-center of the figure, even immediately following the disaster. Looking at that point as a start point, we can observe that the arrows spread out from the point; it means that there is a high probability that a disaster or other event requiring urgent evacuation is occurring nearby the point. In **Fig. 8(b)**, it is apparent that the route used for evacuation runs from the bottom-center to the upper-right, but the arrows near the center-right are small compared with other evacuation routes. Therefore, it is understood that the average movement velocity in this area is low.

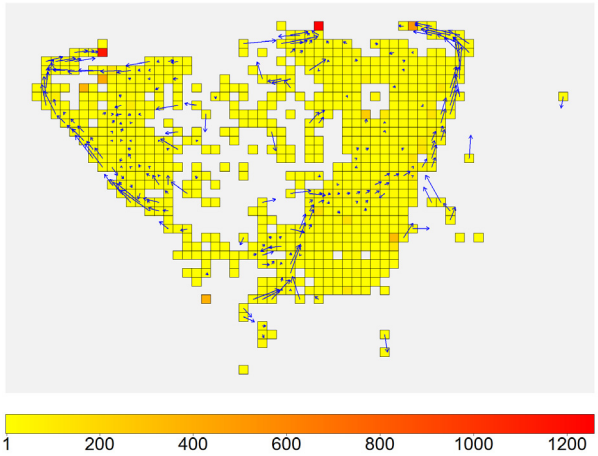
Although **Fig. 8(b)** shows the movement of all individuals, we can also consider a visualization using the status attributes assigned to the simulation data. **Fig. 9(a)** is a visualization that is limited to individuals whose status is listed as "evacuating" in **Fig. 8(b)**. Their evacuation routes can be observed more clearly compared with **Fig. 8(b)**. Conversely, **Fig. 9(b)** is a visualization limited to those whose status is "engaged in firefighting or rescue activities." Compared with evacuees, their movements are minimal, and it is apparent that activities are limited at specific spaces.

5. Performance Evaluation

We measure query execution time in SciDB and evaluate the performance. Since we used extraction and integration operations in SciDB in the analysis examples in Section 4, we here investigate the execution time for queries related to those operations. Below, we explain the method used for measuring execution time. Next, we introduce each query used and the results of each. Finally, we consider the characteristics and tendencies of SciDB demonstrated by the experiments.

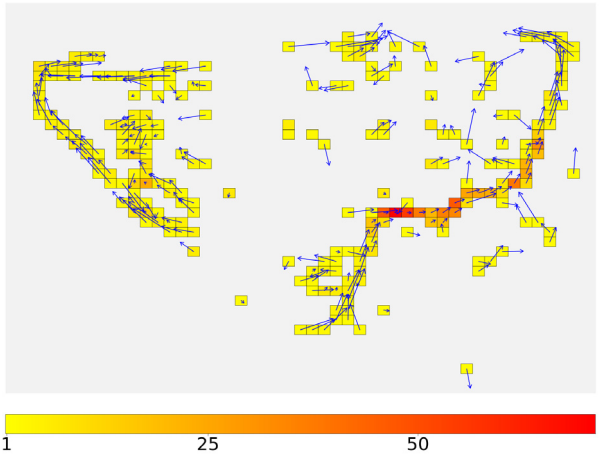


(a) Time $t = 5$ (min)

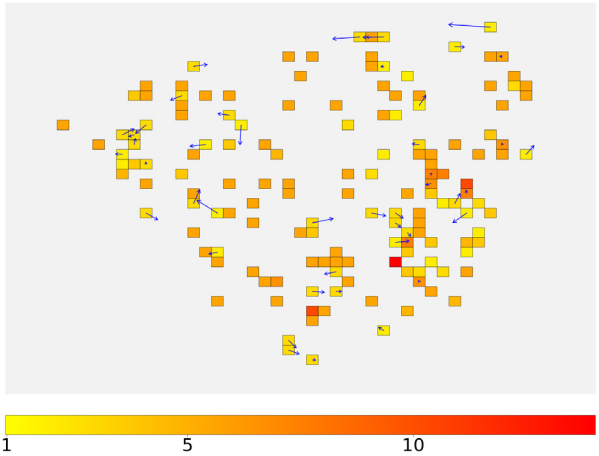


(b) Time $t = 60$ (min)

Fig. 8. Visualization examples.



(a) Status = "evacuating"



(b) Status = "participating in firefighting or rescue activities"

Fig. 9. Visualization by status (time $t = 60$).

5.1. Experiment Environment

The experiment environment uses two types of servers: one coordinator server (CPU: Intel Xeon E5-2637 v4 @ 3.50 GHz × 2; RAM: 128 GB; OS: Ubuntu 14.04 LTS 64-bit) and three worker servers (CPU: Intel Xeon CPU E5620 @ 2.40 GHz × 2; RAM: 32 GB; OS: Ubuntu 14.04 LTS 64-bit). The version of SciDB is 16.9. The coordinator server runs one instance of SciDB; the worker servers each runs three instances, producing parallel processing using ten instances.

5.2. Measurement Method

Queries are issued to SciDB using the R interface, and the execution time is measured. Since the R language has the system.time function for measuring the execution time of a function, runtime is measured using it. The measurement target is query execution time in SciDB. When the R interface is used, the query results are transformed into an R data frame; we omit the time taken for this conversion. Each query is executed 1,000 times, and the average execution time is presented.

The array data used in this experiment is as follows:

```
array <count:int64> [t=0:14400,41,0, x=0:882,41,0,
 y=0:449,41,0, status=1:14,14,0]
```

Before the experiment, the simulation data in Section 2 was preprocessed by the R language and stored in SciDB as a multidimensional array. The size of the spatial grid is 5 m × 5 m, and the time step is 0.1 min. The chunk size is 14 only for status; for other dimensions, the chunk sizes are set equally. The number of elements in the array is approximately 10 million.

In addition, to measure the execution time when the size of the array chenges, arrays with different sizes were prepared using the between operation. The query is as follows:

```
store(between(array, 0, null, null, null,
                [0-14400], null, null, null),
              array_t0-[0-14400])
```

The lower limit of t is set at 0, and the upper limit is set from 0 to 14,400 by the intervals of 200.

5.3. Queries and Execution Times

5.3.1. Changing Dimension Range

First, we examine the execution time when the range of target dimension is altered.

a. **Query 1 (between operation):** As data analysis in SciDB is performed based on dimensions, the between function, which extracts elements within a specified dimensional range, is the basis for analysis. In Query 1,

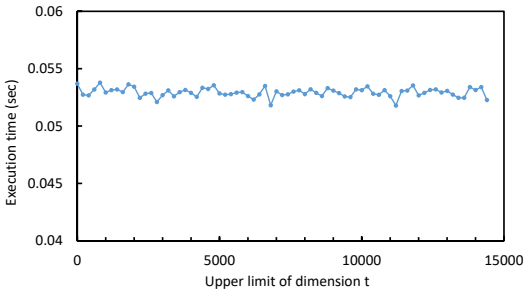


Fig. 10. Execution time of Query 1.

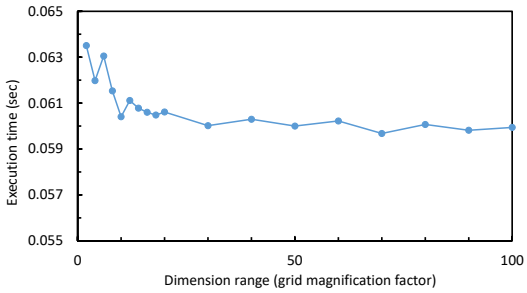


Fig. 11. Execution time of Query 2.

the change in execution time when the range of specified dimension is altered is examined. Range specification is performed only for time step t, and the status of individuals and the spatial coordinates (x,y) are not specified. The lower limit of t was set at 0, while the upper limit was set at from 0 to 14,400 in increments of 200. The execution time for the following query below is measured.

```
between(array, 0, null, null, null, [0-14400],
        null, null, null)
```

The result is shown in **Fig. 10**. It shows that the dimensional range has no impact for the between function, that the execution time is relatively constant at approximately 0.05 s, and that stable queries are possible.

b. **Query 2 (regrid operation):** As stated in Subsection 4.2.2, changing the grid granularity frequently appears in spatio-temporal simulation data analysis. In SciDB, the regrid operation is used to change the grid granularity, and the specified dimensional range corresponds to the grid magnification factor. Since we can consider various types of queries specifying different dimensional ranges, we examine how dimensional range and execution time are connected.

Using Query 2, the relationship between the dimensional range (magnification factor) and the execution time in the regrid operation is examined. The summation function sum is used as the aggregation function, and the size of the spatial grid (x,y) for each time step is changed. First, we create array data for $t = 0$ using the following query:

```
store(slice(array,t,0),array_t0)
```

The number of elements in the array is 3,642. For an array with $t = 0$, the execution time for the following query is measured.

```
regrid(array_t0, [2-100], [2-100], 1,
       sum(count))
```

The result of the experiment is shown in **Fig. 11**. The execution time for the regrid operation is approximately 0.06 s. By increasing the dimensional range (grid magnification factor), we can find some decrease in execution time.

5.3.2. Changing Array Size

Next, execution time is examined for different array sizes. When large-scale spatio-temporal simulation data is managed in SciDB, the number of elements and array dimensions will become extremely large. By examining the relationship between the array dimensions, the number of elements, and execution time, it is possible to provide better estimation of how much time is necessary for executing queries when managing large-scale array data.

Different array sizes of are prepared beforehand using the between function, and queries regarding these array sizes are executed. The upper limit t and number of elements for these arrays, as well as the execution time, are recorded as results. Then, the relationship between the upper limit t, the number of elements, and each execution time is examined.

a. **Query 3 (between operation):** In Query 3, the relationship between array size and execution time is examined in the between operation. The dimension range is $[x = 300 : 500, y = 200 : 400]$. The execution time for the following query is measured for each array size:

```
between(array_t0-[0-14400], null, 300, 200,
        null, null, 500, 400, null)
```

The result is shown in **Fig. 12**. The processing of the between operation has no relationship to the upper limit of time step t or the number of elements, with processing time fairly constant at approximately 0.05 s.

b. **Query 4 (regrid operation):** Query 4 examins the relationship between array size and execution time in the regrid operation. The sum function is used as the aggregation function. The magnification factor for the spatial grid (x,y) is set at $10\times$ for one side. For the following query, execution time is measured for arrays by varying sizes:

```
regrid(array_t0-[0-14400],1 ,10, 10, 1,
       sum(count))
```

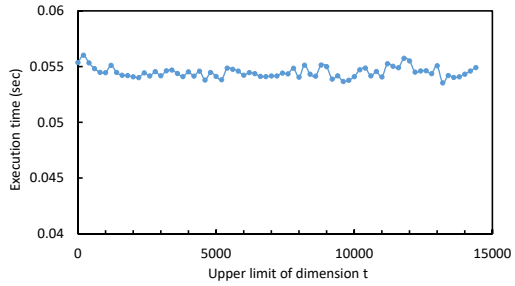
The result of the experiment is shown in **Fig. 13(a)**. As the execution time increases linearly with time step t, it is clear that a linear relationship exists between the size of the dimension and the execution time. In addition, based on **Fig. 13(b)**, the upper limit that can be executed within 1 s is about 5 million.

An experiment was also conducted with the aggregate operation, but an almost identical result was obtained. Therefore, we omit the result in this paper.

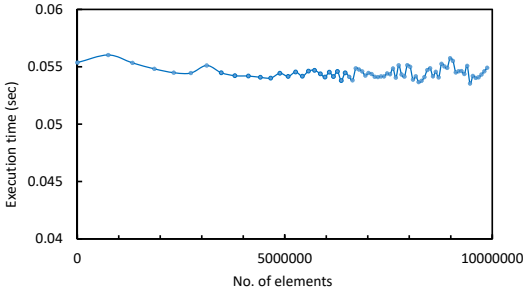
5.4. Discussion

5.4.1. Chunk Size

The results in **Fig. 13(a)** show that the query time increases with the size of the dimension in the array. Since each chunk has a constant length, the number of chunks increases in proportional to the size of the given dimen-

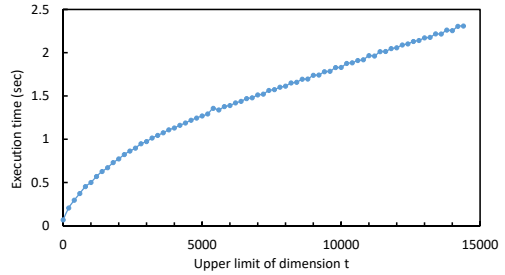


(a) Varying upper limit for t

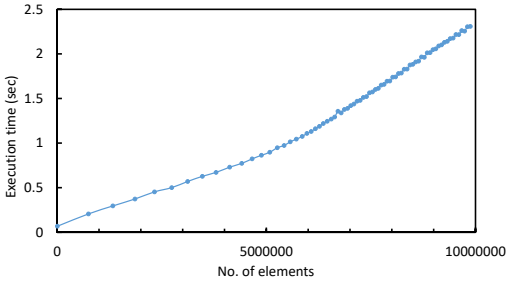


(b) Varying no. of elements

Fig. 12. Execution time of Query 3.



(a) Varying upper limit for t



(b) Varying no. of elements

Fig. 13. Execution time of Query 4.

sion. Based on the result, we can say that the operation processing time in SciDB is greatly influenced by the size of the dimension and the number of chunks. Therefore, when using SciDB, it is necessary to consider the properties of the target array data when determining the chunk size.

In the visualization of the evacuation simulation data used in this analysis, the dimension t is generally a single value, while the ranges of the spatial dimension (x, y) and an individual's status take all the possible ranges. Therefore, chunk size would be more effective from the standpoint of visualization as t becomes smaller and as x, y, and status become larger. Conversely, with analysis requests such as an overview of the number of individuals by status, it is considered that single status value is specified and that the status chunk size therefore should be small. In addition, as it is difficult to consider extracting single spatial grids and performing aggregation by using time steps, it would be appropriate the chunk size for this data to take large size for (x, y).

5.4.2. Execution Time

The result in **Fig. 11** shows that, for the regrid operation, the smaller the dimensional range, the larger the execution time. As the number of elements is fixed, the number of executions for the aggregation function sum is considered to be unchanging, regardless of changes to the dimension range; the fact that the number of array elements obtained as a result is still relatively large is related to the overhead.

The regrid function is used to change the granularity of the spatio-temporal grid. To achieve interactivity, it is important to determine how executions can be handled within 1 s. The result in **Fig. 13(b)** shows that, with 5 million elements, the regrid operation can integrate the grid in less than 1 s. Considering the example from visualization analysis, the range that a user will see at one time is a spatial grid at a single or several time steps. Therefore, grid integration should be performed whenever necessary, as interactivity is sufficiently supported even with 5 million elements. As stated above, changing how chunks are created can accelerate query processing. Consequently, interactivity is sufficiently supported even for spatial grids with an even larger number of elements.

6. Conclusion and Future Work

In this paper, we proposed the use of the SciDB array-oriented DBMS to analyze evacuation simulation data in disasters. SciDB's features include chunk segmentation and parallel distributed processing, which enable it to achieve high-speed dimension-based processing. As an example of evacuation simulation data, an analysis was conducted in multidimensional arrays using spatio-temporal dimensions. Experiments were conducted to measure the execution time for queries in SciDB needed for analysis, and these characteristics were analyzed.

Future topics to be examined include verification experiments using larger-scale simulation data and the development of more-advanced analysis examples and queries that take other scenarios into consideration. Furthermore, from a systemic perspective, the development of interactive system technology including data visualization will be important. When taking the general user into consideration, an interface that users can operate intuitively, without the use of SciDB or the R language that has so far served as its interface, would be considered desirable.

Acknowledgements

We would like to thank Osaragi Laboratory at Tokyo Institute of Technology for providing the evacuation simulation data. Part of this study was funded by the CREST "Creation of Innovative Earthquake and Tsunami Disaster Reduction Big Data Analysis Foundation by Cooperation of Large-Scale and High-Resolution

Numerical SImulatins and Data Assimilations" and by Grant-in-Aid for Scientific Research (16H01722).

References:

[1] "The architecture and motivation for Paradigm4's SciDB," Technical report, Paradigm4, 2016.

[2] Paul G. Brown, "Overview of SciDB: Large scale array storage, processing and analysis," In Proc. ACM SIGMOD, pp. 963–968, 2010, doi: 10.1145/1807167.1807271.

[3] H. Lustosa, F. Porto, P. Valduriez, and P. Blanco, "Database system support of simulation data," Proc. VLDB Endow. (PVLDB), Vol.9, No.13, pp. 1329–1340, 2016, doi: 10.14778/3007263.3007271.

[4] T. Osaragi and T. Oki, "Wide-area evacuation simulation incorporating rescue and firefighting by local residents," Journal of Disaster Research, Vol.12, No.2, pp. 296–310, 2017, doi: 10.20965/jdr.2017.p0296.

[5] Paradigm4: Creators of SciDB a computational DBMS, http://www.paradigm4.com/ [accessed October 31, 2017]

[6] M. Stonebraker, P. Brown, J. Becla, and D. Zhang, "SciDB: A database management system for applications with complex analytics," IEEE Computational Science & Engineering, Vol.15, No.3, pp. 54–62, 2013, doi: 10.1109/MCSE.2013.19.

[7] J. Zhao, Y. Ishikawa, Y. Wakita, and K. Sugiura, "Difference operators in simulation data warehouses," Journal of Disaster Research, Vol.12, No.2, pp. 347–354, 2017, doi: 10.20965/jdr.2017.p0347.

[8] J. Zhao, K. Sugiura, Y. Wang, and Y. Ishikawa, "Simulation data warehouse for integration and analysis of disaster information," Journal of Disaster Research, Vol.11, No.2, pp. 255–264, 2016, doi: 10.20965/jdr.2017.p0347.

Name:
Yusuke Kawai

Affiliation:
Master Student, Graduate School of Informatics, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

Brief Career:
2017- Master Student, Graduate School of Informatics, Nagoya University

Academic Societies & Scientific Organizations:
• Database Society of Japan (DBSJ)

Name:
Jing Zhao

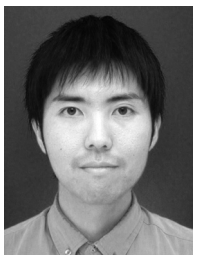
Affiliation:
Ph.D. Candidate, Graduate School of Information Science, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

Brief Career:
2013- Master Student, Graduate School of Information Science, Nagoya University
2015-Ph.D. Candidate, Graduate School of Information Science, Nagoya University

Selected Publications:
• "Difference Operators in Simulation Data Warehouses," Journal of Disaster Research, Vol.12, No.2, pp. 347-354, March 2017.

Academic Societies & Scientific Organizations:
• Database Society of Japan (DBSJ)



Name:
Kento Sugiura

Affiliation:
Ph.D. Candidate, Graduate School of Information Science, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

Brief Career:
2013- Master Student, Graduate School of Infor. Sci., Nagoya Univ.
2015- Ph.D. Candidate, Graduate School of Infor. Sci., Nagoya Univ.

Selected Publications:
• "Grouping Methods for Pattern Matching over Probabilistic Data Streams," IEICE Transactions on Information and Systems, Vol. E100-D, No. 4, pp. 718-729, April 2017.

Academic Societies & Scientific Organizations:
• Information Processing Society of Japan (IPSJ)
• Database Society of Japan (DBSJ)

Name:
Yoshiharu Ishikawa

Affiliation:
Professor, Graduate School of Informatics, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

Brief Career:
2003- Associate Professor, University of Tsukuba
2006- Professor, Nagoya University

Selected Publications:
• "Probabilistic Range Querying over Gaussian Objects," IEICE Transactions on Information and Systems, Vol.E97-D, No.4, pp. 694-704, April 2014.

Academic Societies & Scientific Organizations:
• ACM, IEEE Computer Society, IPSJ, IEICE, DBSJ

Name:
Yukiko Wakita

Affiliation:
Research Associate, Graduate School of Informatics, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

Brief Career:
2014- Researcher, Nagoya University
2016- Research Associate, Nagoya University

Selected Publications:
"Traffic Network Design by Cellular Automaton-based Traffic Simulator," Computer Assisted Methods in Engineering and Science, Vol.22, No.1, pp. 51-61, 2015.

Academic Societies & Scientific Organizations:
• Information Processing Society of Japan (IPSJ)