(12) BẢN MÔ TẢ SÁNG CHẾ THUỘC BẰNG ĐỘC QUYỀN SÁNG CHẾ

(19) **Cộng hòa xã hội chủ nghĩa Việt Nam (VN)**
CỤC SỞ HỮU TRÍ TUỆ

(11) **1-0031543**

(51)[8] **F21V 29/00**

(13) **B**

(21) 1-2018-04508

(22) 11/10/2018

(45) 25/04/2022 409

(43) 25/01/2019 370A

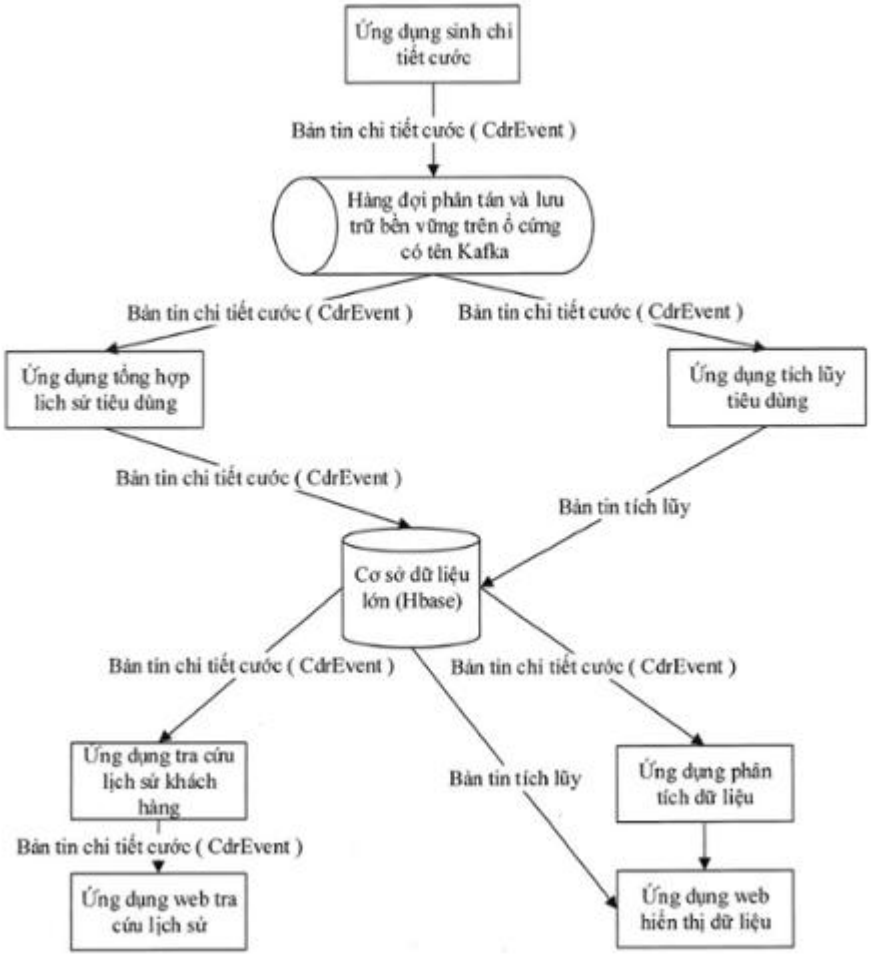
(73) Tập đoàn Công nghiệp - Viễn thông Quân đội (VIETTEL) (VN)
Số 1 Trần Hữu Dực, Mỹ Đình 2, quận Nam Từ Liêm, thành phố Hà Nội

(72) Trịnh Văn Hùng (VN); Nguyễn Đức Hải (VN).

(74) Công ty Luật TNHH quốc tế BMVN (BMVN INTERNATIONAL LLC)

(54) PHƯƠNG PHÁP GIẢM KÍCH THƯỚC BẢN TIN CHỨA DANH SÁCH CÁC ÁNH XẠ KHÓA - GIÁ TRỊ THEO TẦN SUẤT KIỂU DỮ LIỆU KHÓA - GIÁ TRỊ TRONG XỬ LÝ DỮ LIỆU THỜI GIAN THỰC

(57) Sáng chế đề xuất phương pháp giảm kích thước bản tin chứa danh sách các ánh xạ khóa - giá trị theo tần suất kiểu dữ liệu khóa - giá trị trong xử lý dữ liệu thời gian thực, bao gồm các bước: a) xây dựng bảng mã các kiểu dữ liệu cần chuyển đổi của khóa - giá trị; b) xác định kiểu dữ liệu chuyển đổi cho khóa - giá trị trong bản tin; c) gom các cặp khóa - giá trị có cùng kiểu định nghĩa vào một nhóm; d) lưu các cặp khóa - giá trị đã được chuyển đổi theo nhóm vào vùng nhớ đệm; e) sao chép từ vùng nhớ đệm vào mảng byte kết thúc việc serialization từ đối tượng sang mảng byte; và f) thực hiện chuyển mảng byte sang kiểu bản tin bằng cách xác định kiểu dữ liệu dựa vào giá trị của byte.


Ứng dụng sinh chi tiết cước
Bản tin chi tiết cước (CdrEvent)
Hàng đợi phân tán và lưu trữ bền vững trên ổ cứng có tên Kafka
Bản tin chi tiết cước (CdrEvent)
Bản tin chi tiết cước (CdrEvent)
Ứng dụng tổng hợp lịch sử tiêu dùng
Ứng dụng tích lũy tiêu dùng
Bản tin chi tiết cước (CdrEvent)
Bản tin tích lũy
Cơ sở dữ liệu lớn (Hbase)
Bản tin chi tiết cước (CdrEvent)
Bản tin chi tiết cước (CdrEvent)
Ứng dụng tra cứu lịch sử khách hàng
Bản tin tích lũy
Ứng dụng phân tích dữ liệu
Bản tin chi tiết cước (CdrEvent)
Ứng dụng web tra cứu lịch sử
Ứng dụng web hiển thị dữ liệu

Lĩnh vực kỹ thuật được đề cập

Sáng chế đề cập đến phương pháp giảm kích thước bản tin chứa danh sách các ánh xạ khóa - giá trị (map key - value) trong việc phát triển các hệ thống công nghệ thông tin và viễn thông trên nền tảng ngôn ngữ lập trình Java, chẳng hạn như trong việc truyền nhận các bản tin chứa ánh xạ khóa - giá trị trong ứng dụng công nghệ thông tin, lưu trữ đối tượng chứa khóa - giá trị trong cơ sở dữ liệu và tệp dữ liệu.

Tình trạng kỹ thuật của sáng chế

Các kĩ thuật "chuyển đổi đối tượng hoặc bản tin sang dạng mảng byte" (serialization) được hỗ trợ trong nhiều thư viện chẳng hạn như thư viện Protocol Buffer (Protobuf), Message Pack. Tuy nhiên, để thực hiện kỹ thuật serialization đạt được kích thước nhỏ nhưng vẫn được xử lý nhanh và việc thêm mới kiểu "giá trị" dữ liệu linh hoạt là chưa có. Hiện nay, các thư viện Protocol Buffer, Message Pack chỉ hỗ trợ kỹ thuật serialization danh sách ánh xạ khóa - giá trị ở dạng định nghĩa trước kiểu dữ liệu của khóa - giá trị. Mỗi lần thay đổi kiểu dữ liệu phải tạo lại mã (code), dẫn đến khi có thay đổi về kiểu dữ liệu thì phải thực hiện cập nhật mã mới lên hệ thống. Ngoài ra, các thư viện nêu trên cũng không hỗ trợ đồng thời các kiểu dữ liệu số (Byte, Short, Integer, Long), kiểu dữ kiệu thời gian (Date), kiểu dữ liệu xâu (String), kiểu dữ liệu mảng byte (byte array) trong kiểu "giá trị". Thư viện truyền thống Java hỗ trợ serialization tất cả các kiểu dữ liệu thực thi giao diện Serializable, tuy nhiên tốc độ xử lý còn chậm và kích thước bản tin lớn không hiệu quả trong hệ thống yêu cầu thời gian thực.

Mục đích của sáng chế

Nhằm khắc phục các thiếu sót nêu trên, các tác giả sáng chế đã nghiên cứu phương pháp giảm kích thước dữ liệu các bản tin chứa các ánh xạ khóa - giá trị theo tần suất xuất hiện của kiểu cặp giá trị khóa - giá trị.

Bản chất kỹ thuật của sáng chế

Sáng chế đề xuất phương pháp giảm kích thước bản tin chứa danh sách các ánh xạ khóa - giá trị theo tần suất kiểu dữ liệu khóa - giá trị trong xử lý dữ liệu thời gian thực, bao gồm các bước sau:

a) xây dựng bảng mã các kiểu dữ liệu cần chuyển đổi của khóa - giá trị:

ai) đánh mã các thuộc tính dữ liệu;

aii) xây dựng giao diện cho phép đọc các thuộc tính này;

aiii) định nghĩa kiểu dữ liệu cần chuyển đổi;

b) xác định kiểu dữ liệu chuyển đổi cho khóa - giá trị trong bản tin:

bi) xác định kiểu dữ liệu khóa;

bii) xác định kiểu dữ liệu giá trị;

biii) kết hợp khóa và giá trị ta xác định được kiểu khóa - giá trị;

c) gom các cặp khóa - giá trị có cùng kiểu định nghĩa vào một nhóm:

ci) duyệt tất cả các cặp khóa - giá trị;

cii) gom các cặp khóa - giá trị cùng kiểu định nghĩa vào 1 nhóm;

d) lưu các cặp khóa - giá trị đã được chuyển đổi theo nhóm vào vùng nhớ đệm:

di) ghi kiểu dữ liệu cặp khóa - giá trị của nhóm;

dii) ghi số lượng các cặp có cùng kiểu;

diii) ghi dữ liệu các cặp khóa - giá trị trong nhóm;

e) sao chép từ vùng nhớ đệm vào mảng byte kết thúc việc serialization từ đối tượng sang mảng byte; và

f) thực hiện chuyển mảng byte sang kiểu bản tin bằng cách xác định kiểu dữ liệu dựa vào giá trị của byte:

fi) xác định kiểu dữ liệu của nhóm;

fii) xác định số lượng cặp trong nhóm;

fiii) đọc các cặp khóa - giá trị trong nhóm cho đủ số lượng cặp;

fiv) đọc lặp lại cho đến hết mảng byte.

Như vậy, phương pháp theo sáng chế là sự số hóa các thuộc tính mà con người dễ đọc hiểu sang số nguyên giới hạn kiểu short. Tiếp theo chuyển đổi kiểu dữ liệu khóa - giá trị sang kiểu dữ liệu tối giản nhất mà không mất thông tin dữ liệu. Từ đó ta xây dựng được 30 kiểu dữ liệu có khả năng đáp ứng nhu cầu thường dùng. Cuối cùng là gom các cặp khóa - giá trị vào cùng 1 nơi để hạn chế việc ghi lại thường xuyên kiểu giá trị của khóa - giá trị.

Mô tả vắn tắt các hình vẽ

Hình 1 là lưu đồ thể hiện luồng chuyển đổi bản tin sang mảng byte.

Hình 2 là lưu đồ thể hiện luồng chuyển đổi mảng byte sang bản tin.

Hình 3 là sơ đồ khối thể hiện một ví dụ thực hiện phương pháp theo sáng chế trong hệ thống tính cước thời gian thực của Viettel (Viettel Online Charging System - VOCS).

Hình 4 là biểu đồ so sánh các kích thước kiểu dữ liệu số thu được trong Protocol Buffer, Java và theo phương pháp của sáng chế.

Hình 5 là biểu đồ so sánh các kích thước kiểu dữ liệu số được đưa vào giá trị kiểu mảng byte thu được trong Protocol Buffer, Java và theo phương pháp của sáng chế.

Hình 6 là biểu đồ so sánh tốc độ giữa phương pháp theo sáng chế, Protocol Buffer, Java trên 2 kiểu dữ liệu Byte Array và số Long.

Mô tả chi tiết sáng chế

Phương pháp theo sáng chế được thực hiện như sau:

Thực hiện chuyển bản tin sang dạng mảng byte:

Khi khởi tạo ứng dụng, thực hiện việc chuyển bản tin sang dạng mảng byte theo các bước sau:

- Bước 1: Đánh mã các thuộc tính dữ liệu, sau đó xây dựng giao diện cho phép đọc động các thuộc tính này. Khi thay đổi, chỉ cần đọc lại mà không cần khởi động lại ứng dụng. Bảng 1 thể hiện định nghĩa các thuộc tính được số hóa:

Bảng 1: Định nghĩa các thuộc tính

Thuộc tính	Mã	Kiểu giá trị
TELECOM_SERVICE_ID	0	Integer
CALLING_NUMBER	1	Long
CALLED_NUMBER	2	Long
EVENT_BEGIN_TIME	3	Long
CELL_ID	4	Integer
BALANCE_01	5	Long
PRE_BALANCE_01	6	Long
CHARGE_01	7	Long
BALANCE_02	8	Long
PRE_BALANCE_02	9	Long
CHARGE_02	10	Long

Như vậy kiểu dữ liệu khóa (key) chỉ là số nguyên giới hạn cho hệ thống đang để ở mức số nguyên 2 byte (Short). Với mức này, ta biểu diễn số lượng key bằng số lượng số nguyên của kiểu Short.

Bảng 2: Chuyển đổi kiểu giá trị

Giá trị	Kiểu dữ liệu gốc	Kiểu dữ liệu chuyển đổi
100	Integer(4byte)	Byte(1byte)
512	Integer(4byte)	Short(2byte)
200000	Long(8byte)	Integer(4byte)

- Bước 2: Định nghĩa kiểu dữ liệu cần chuyển đổi của khóa - giá trị như được thể hiện trong Bảng 3. Khi cần bổ sung thêm kiểu dữ liệu mới, có thể sử dụng các định nghĩa được nêu trong Bảng 3 này. Bộ 30 kiểu dữ liệu cần chuyển đổi được định nghĩa trong Bảng 3 này có thể bao quát được hầu hết các kiểu dữ liệu thường dùng. Các kiểu dữ liệu khác cần được chuyển sang dạng mảng byte là có thể áp dụng được.

Bảng 3: Định nghĩa kiểu dữ liệu Khóa - Giá trị

Kiểu Khóa gốc	Giá trị của Khóa	Kiểu Giá trị gốc	Giá trị	Kiểu chuyển đổi Khóa - giá trị	Mã kiểu Khóa - giá trị	Ý nghĩa
Short	12	Short	32	BS2B	0	Key dạng Byte, Value Short chuyển Byte
Short	12	Short	320	BS2S	1	Key dạng Byte, Value Short chuyển Short
Short	12	Integer	32	BI2B	2	Key dạng Byte, Value Integer chuyển Byte
Short	12	Integer	512	BI2S	3	Key dạng Byte, Value Integer chuyển Short
Short	12	Integer	66666	BI2I	4	Key dạng Byte, Value Integer chuyển Integer
Short	12	Long	32	BL2B	5	Key dạng Byte, Value Long chuyển Byte
Short	12	Long	512	BL2S	6	Key dạng Byte, Value Long chuyển Short
Short	12	Long	66666	BL2I	7	Key dạng Byte, Value Long chuyển Integer

Kiểu Khóa gốc	Giá trị của Khóa	Kiểu Giá trị gốc	Giá trị	Kiểu chuyển đổi Khóa - giá trị	Mã kiểu Khóa - giá trị	Ý nghĩa
Short	12	Long	5000000000	BL2L	8	Key dạng Byte, Value Long chuyển Long
Short	12	Date	2018/24/05 00:00:00	BD2L	9	Key dạng Byte, Value Date chuyển Long
Short	12	String	"0975998639"	BBS	10	Key dạng Byte, Value String với độ dài kiểu Byte
Short	12	String	Xâu có độ dài > 256	BSS	11	Key dạng Byte, Value String với độ dài kiểu Short
Short	512	Short	32	SS2B	12	Key dạng Short, Value Short chuyển Byte
Short	512	Short	320	SS2S	13	Key dạng Short, Value Short chuyển Short
Short	512	Integer	32	SI2B	14	Key dạng Short, Value Integer chuyển Byte
Short	512	Integer	512	SI2S	15	Key dạng Short, Value Integer chuyển Short
Short	512	Integer	66666	SI2I	16	Key dạng Short, Value Integer chuyển Integer
Short	512	Long	32	SL2B	17	Key dạng Short, Value Long chuyển Byte
Short	512	Long	512	SL2S	18	Key dạng Short, Value Long chuyển Short
Short	512	Long	66666	SL2I	19	Key dạng Short, Value Long chuyển Integer
Short	512	Long	5000000000	SL2L	20	Key dạng Short, Value Long chuyển Long
Short	512	Date	2018/24/05 00:00:00	SD2L	21	Key dạng Short, Value Date chuyển Long

Kiểu Khóa gốc	Giá trị của Khóa	Kiểu Giá trị gốc	Giá trị	Kiểu chuyển đổi Khóa - giá trị	Mã kiểu Khóa - giá trị	Ý nghĩa
Short	512	String	"0975998639"	SBS	22	Key dạng Short, Value String với độ dài kiểu Byte
Short	512	String	Xâu có độ dài > 256	SSS	23	Key dạng Short, Value String với độ dài kiểu Short
Short	12	Byte Array	byte[100]	BBA	24	Key dạng Byte, Value Byte Array với độ dài kiểu Byte
Short	12	Byte Array	byte[512]	BSA	25	Key dạng Byte, Value Byte Array với độ dài kiểu Short
Short	512	Byte Array	byte[100]	SBA	26	Key dạng Short, Value Byte Array với độ dài kiểu Byte
Short	512	Byte Array	byte[512]	SSA	27	Key dạng Short, Value Byte Array với độ dài kiểu Short
Short	12	Byte	32	BB2B	28	Key dạng Byte, Value Byte chuyển Byte
Short	512	Byte	32	SB2B	29	Key dạng Short, Value Byte chuyển Byte

Phương pháp xác định kiểu dữ liệu chuyển đổi cho khóa - giá trị trong bản tin. Đối với khóa ta có bảng chuyển đổi sau

Bảng 4: Xác định kiểu dữ liệu khóa

Kiểu Khóa gốc	Giá trị của Khóa	Kiểu chuyển đổi Khóa	Ý nghĩa

Short	Giới hạn kiểu Byte	B	Key dạng Byte
Short	Giới hạn kiểu Short	S	Key dạng Short

Đối với giá trị ta có bảng chuyển đổi sau

Bảng 5: Định nghĩa kiểu dữ liệu Giá trị

Kiểu Giá trị gốc	Giá trị	Kiểu chuyển đổi Giá trị	Ý nghĩa
Byte	Giới hạn kiểu Byte	B2B	Value Byte chuyển Byte
Short	Giới hạn kiểu Byte	S2B	Value Short chuyển Byte
Short	Giới hạn kiểu Short	S2S	Value Short chuyển Short
Integer	Giới hạn kiểu Byte	I2B	Value Integer chuyển Byte
Integer	Giới hạn kiểu Short	I2S	Value Integer chuyển Short
Integer	Giới hạn kiểu Integer	I2I	Value Integer chuyển Integer
Long	Giới hạn kiểu Byte	L2B	Value Long chuyển Byte
Long	Giới hạn kiểu Short	L2S	Value Long chuyển Short
Long	Giới hạn kiểu Integer	L2I	Value Long chuyển Integer
Long	Giới hạn kiểu Long	L2L	Value Long chuyển Long
Date	Chuyển sang kiểu Long	D2L	Value kiểu Date chuyển Long
String	Độ dài giới hạn Byte	BS	Value kiểu String có độ dài kiểu Byte
String	Độ dài giới hạn Short	SS	Value kiểu String có độ dài kiểu Short
ByteArray	Độ dài giới hạn Byte	BA	Value kiểu Byte Array có độ dài kiểu Byte
ByteArray	Độ dài giới hạn Short	SA	Value kiểu Byte Array có độ dài kiểu Short

Kết hợp khóa và giá trị ta được xác định kiểu khóa - giá trị.

Bảng 6 dưới đây thể hiện sự chuyển đổi kiểu dữ liệu mẫu cho Khóa - Giá trị.

Bảng 6: Chuyển đổi kiểu dữ liệu mẫu cho Khóa - Giá trị

STT	Khóa	Kiểu Khóa	Giá trị	Kiểu giá trị	Kiểu Khóa - giá trị
1	12	Short	15	Integer	BI2B - 2
2	13	Short	99	Integer	BI2B - 2
3	14	Short	103	Integer	BI2B - 2
4	16	Short	300	Integer	BI2S - 3
5	17	Short	350	Integer	BI2S - 3
6	18	Short	400	Integer	BI2S - 3
7	19	Short	byte[20]	Byte Array	BBA - 24
8	20	Short	byte[40]	Byte Array	BBA - 24
10	21	Short	byte[256]	Byte Array	BSA - 25
11	22	Short	byte[300]	Byte Array	BSA - 25

Từ bảng nêu trên ta thấy:

Các cặp {12:15},{13:99},{14:103} có cùng kiểu BI2B (2)

Các cặp {16:300},{17:350},{18:400} có cùng kiểu BI2S (3)

Các cặp {19: byte[20]},{20: byte[40]} có cùng kiểu BBA (24)

Các cặp {21: byte[256]},{22: byte[300]} có cùng kiểu BSA (25)

Khi chuyển đổi sang mảng byte ta thực hiện lưu [kiểu cặp khóa - giá trị][số lượng cặp][Lần lượt giá trị các cặp khóa - giá trị đã được tính kiểu].

Với "số lượng cặp" ta thực hiện lưu như sau. Bit đầu tiên của byte là 1 thì độ dài của cặp số lượng là 2 byte (cần đọc thêm 1 byte nữa). Nếu bit đầu tiên của byte là 0 thì độ dài của cặp chỉ là 1 byte.

Bảng 7: Bảng lưu mẫu cho Khóa - Giá trị

	Dữ liệu	Số Byte	Ý nghĩa
Kiểu	2	1	1 byte lưu kiểu
Số lượng	3	1	1 byte lưu số lượng Key-Value
Key - Value	1215	1+1	1 byte lưu key - 1 byte lưu value
Key - Value	1399	1+1	1 byte lưu key - 1 byte lưu value

Key - Value	14103	1+1	1 byte lưu key - 1 byte lưu value
Kiểu	3	1	1 byte lưu kiểu
Số lượng	3	1	1 byte lưu số lượng Key-Value
Key - Value	16300	1+2	1 byte lưu key - 2 byte lưu value
Key - Value	17350	1+2	1 byte lưu key - 2 byte lưu value
Key - Value	18400	1+2	1 byte lưu key - 2 byte lưu value
Kiểu	24	1	1 byte lưu kiểu
Số lượng	2	1	1 byte lưu số lượng Key-Value
Key - Value	1920byte[20]	1+1+20	1 byte lưu key - 1 byte lưu độ dài array - 20 byte lưu byte array
Key - Value	2040byte[40]	1+1+40	1 byte lưu key - 1 byte lưu độ dài array - 40 byte lưu byte array
Kiểu	25	1	1 byte lưu type
Số lượng	2	1	1 byte lưu số lượng Key-Value
Key - Value	21256byte[256]	1+2+256	1 byte lưu key - 2 byte lưu độ dài array - 256 byte lưu byte array
Key - Value	22300byte[300]	1+1+300	1 byte lưu key - 2 byte lưu độ dài array - 300 byte lưu byte array

Phương pháp lưu tổng quát.

[kiểu cặp giá trị 1][số lượng N1][giá trị khóa 1 (key), giá trị 1(value), … giá trị khóa N1 (key), giá trị N1(value)] [kiểu cặp giá trị 2][số lượng N2][giá trị khóa 1 (key), giá trị 1 (value), … giá trị khóa N2 (key), giá trị N2 (value)]…

Đối với kiểu dữ liệu giá trị là mảng byte thì ghi chiều dài mảng trước giá trị. Với phương pháp trên, tác giả đã áp dụng vào phân hệ sinh chi tiết cước và phân hệ phân tích hành vi khách hàng mang lại hiệu quả vượt trội về lưu lượng, hiệu năng và khả năng định nghĩa động dữ liệu.

Khi chuyển đổi bản tin, luồng dữ liệu chuyển đổi bản tin sang byte array được thể hiện trong Hình 1 bao gồm các bước:

- Bước 1: Duyệt tất cả các cặp khóa - giá trị nhóm các cặp có cùng kiểu định nghĩa vào 1 nhóm. Ở bước này ta thu được kiểu dữ liệu giá trị key đã chuyển, giá trị value đã chuyển

- Bước 2: Thực hiện ghi các cặp khóa - giá trị đã được chuyển đổi theo nhóm vào vùng nhớ đệm. Để tăng tốc xử lý lựa chọn các đối tượng cấp phát bộ nhớ nhanh trên Java thông qua giao diện lớp ByteBuffer hoặc Unsafe. Để xử lý tối ưu cho đa luồng ta sử dụng các kĩ thuật lập trình với giao diện lớp ThreadLocal.

+ Ghi kiểu dữ liệu của cặp khóa - giá trị.

+ Ghi số lượng các cặp có cùng kiểu nếu số lượng lớn hơn giới hạn kiểu Byte thì ghi 2 byte. Giới hạn tối đa ở đây là 32767 cặp.

+ Ghi dữ liệu các cặp khóa - giá trị trong nhóm. Nếu kiểu BS2B (0) thì key chiếm 1 byte, value kiểu short nhưng khi lưu chỉ cần 1 byte

Các kiểu giá trị khác tham khảo bảng 8 dưới đây.

Bảng 8: Tính toán số byte lưu trữ

Kiểu	Mã kiểu Khóa - Giá trị	Ý nghĩa	Số byte
BS2B	0	Khóa dạng Byte, Giá trị Short chuyển Byte	1+1
BS2S	1	Khóa dạng Byte, Giá trị Short chuyển Short	1+2
BI2B	2	Khóa dạng Byte, Giá trị Integer chuyển Byte	1+1
BI2S	3	Khóa dạng Byte, Giá trị Integer chuyển Short	1+2
BI2I	4	Khóa dạng Byte, Giá trị Integer chuyển Integer	1+4
BL2B	5	Khóa dạng Byte, Giá trị Long chuyển Byte	1+1
BL2S	6	Khóa dạng Byte, Giá trị Long chuyển Short	1+2
BL2I	7	Khóa dạng Byte, Giá trị Long chuyển Integer	1+4
BL2L	8	Khóa dạng Byte, Giá trị Long chuyển Long	1+8
BD2L	9	Khóa dạng Byte, Giá trị Date chuyển Long	1+8
BBS	10	Khóa dạng Byte, Giá trị	1+1 (độ dài mảng)+ xâu

Kiểu	Mã kiểu Khóa - Giá trị	Ý nghĩa	Số byte
		String với độ dài kiểu Byte	được chuyển sang mảng byte
BSS	11	Khóa dạng Byte, Giá trị String với độ dài kiểu Short	1+2(độ dài mảng)+ xâu được chuyển sang mảng byte
SS2B	12	Khóa dạng Short, Giá trị Short chuyển Byte	2+1
SS2S	13	Khóa dạng Short, Giá trị Short chuyển Short	2+2
SI2B	14	Khóa dạng Short, Giá trị Integer chuyển Byte	2+1
SI2S	15	Khóa dạng Short, Giá trị Integer chuyển Short	2+2
SI2I	16	Khóa dạng Short, Giá trị Integer chuyển Integer	2+4
SL2B	17	Khóa dạng Short, Giá trị Long chuyển Byte	2+1
SL2S	18	Khóa dạng Short, Giá trị Long chuyển Short	2+2
SL2I	19	Khóa dạng Short, Giá trị Long chuyển Integer	2+4
SL2L	20	Khóa dạng Short, Giá trị Long chuyển Long	2+8
SD2L	21	Khóa dạng Short, Giá trị Date chuyển Long	2+8
SBS	22	Khóa dạng Short, Giá trị String với độ dài kiểu Byte	2+1(độ dài mảng)+ xâu được chuyển sang mảng byte
SSS	23	Khóa dạng Short, Giá trị String với độ dài kiểu Short	2+2(độ dài mảng)+ xâu được chuyển sang mảng byte
BBA	24	Khóa dạng Byte, Giá trị Byte Array với độ dài kiểu Byte	1+1(độ dài mảng)+ mảng byte
BSA	25	Khóa dạng Byte, Giá trị Byte	1+2(độ dài mảng)+ mảng

Kiểu	Mã kiểu Khóa - Giá trị	Ý nghĩa	Số byte
		Array với độ dài kiểu Short	byte
SBA	26	Khóa dạng Short, Giá trị Byte Array với độ dài kiểu Byte	2+1(độ dài mảng)+ mảng byte
SSA	27	Khóa dạng Short, Giá trị Byte Array với độ dài kiểu Short	2+2(độ dài mảng)+ mảng byte
BB2B	28	Khóa dạng Byte, Giá trị Byte chuyển Byte	1+1
SB2B	29	Khóa dạng Short, Giá trị Byte chuyển Byte	2+1

- Bước 3: Sao chép vùng nhớ đã ghi vào mảng byte. Kết thúc việc serialization từ đối tượng sang mảng byte.

Thực hiện chuyển mảng byte sang kiểu bản tin

Hình 2 là lưu đồ thể hiện luồng dữ liệu chuyển đổi mảng byte sang bản tin bao gồm các bước sau:

Bước 1: Đọc 1 byte đầu tiên rồi đối chiếu xem kiểu dữ liệu thuộc loại nào.

+ Nếu giá trị là 0 thì thuộc loại BS2B. Tức khóa chiếm 1 byte, giá trị chiếm 1 byte. Đọc 1 byte tiếp theo kiểm tra bit đầu tiên của byte. Nếu là 0 thì số lượng cặp chỉ ở giới hạn byte. Nếu là 1 thì đọc tiếp 1 byte tiếp ta tính ra số lượng cặp khóa - giá trị của kiểu này. Giả sử số lượng cặp đọc ra được là N thì ta thực hiện đọc N lần 1 byte cho khóa, 1 byte cho giá trị. Ta thực hiện đưa vào 1 ánh xạ khóa - giá trị có kiểu khóa số nguyên Integer, kiểu giá trị là Object (Map<Integer,Object>). Tất cả các giá trị thu được từ việc đọc khóa - giá trị.

+ Nếu giá trị là 22 thì thuộc loại SBS. Tức khóa chiếm 2 byte, độ dài của xâu chiếm 1 byte có giá trị X, sau đó X byte thực hiện chuyển đổi về string. Đọc 1 byte tiếp theo kiểm tra bit đầu tiên của byte. Nếu là 0 thì số lượng cặp chỉ ở giới hạn byte. Nếu là 1 thì đọc tiếp 1 byte tiếp ta tính ra số lượng cặp khóa - giá trị của kiểu này. Giả sử số lượng cặp đọc ra được là N thì ta thực hiện đọc N lần 2 byte cho khóa, 1 byte cho độ dài của xâu là X, đọc tiếp X byte ta được giá trị của xâu. Ta thực hiện đưa vào 1 Map<Integer,Object> tất cả các giá trị thu được từ việc đọc khóa - giá trị.

Bước 2: Lặp lại bước 1 cho đến khi hết mảng byte. Kết quả thu được 1 đối tượng của Map chứa các cặp khóa - giá trị đã thực hiện chuyển đổi trước đó.

Ví dụ thực hiện sáng chế

Phương pháp theo sáng chế đã được áp dụng vào ứng dụng sinh chi tiết cước (Trigger), ứng dụng tổng hợp lịch sử tiêu dùng khách hàng (Cust Hist Import), ứng dụng phân tích dữ liệu khách hàng (Customer Behavior Analytics - CBA), ứng dụng tra cứu lịch sử khách hàng (Cust Hist Query), ứng dụng web tra cứu lịch sử (Web Cust Hist), ứng dụng tích lũy tiêu dùng khách hàng (CBA Accumulate), ứng dụng phân tích dữ liệu (CBA Analytics) của hệ thống tính cước thời gian thực (vOCS). Như được thể hiện trên Hình 3.

Bản tin chi tiết cước (Call Detail Record Event - CdrEvent) được thực hiện chuyển sang mảng byte theo phương pháp mô tả trên. Nó đóng vai trò là trung gian truyền nhận giữa các phân hệ, đồng thời là đối tượng được lưu trữ trong cơ sở dữ liệu lớn Hbase và hệ thống Kafka. Từ Trigger sang Kafka, đọc từ Kafka lưu Hbase. Chi tiết các bước thực hiện ứng dụng như sau:

Bước 1:

Xây dựng bảng mã các thuộc tính CDR (Call Detail Record) sử dụng cho tất cả các phân hệ có liên quan. Bảng mã này lưu vào tệp và cơ sở dữ liệu được sử dụng khi khởi động chương trình. Ưu tiên lấy trong cơ sở dữ liệu trước nếu không có thì đọc từ tệp.

Bước 2:

Ứng dụng Trigger nhận các kết quả bản tin tính cước và dữ liệu có liên quan sau đó căn cứ vào cấu hình định nghĩa từng loại CDR sinh bản tin CDREvent tương ứng. Bản tin CdrEvent được thực hiện chuyển sang mảng byte theo phương pháp mô tả trên. Ứng dụng Trigger lưu bản tin trên phân hệ hàng đợi bền vững (Persistent Queue) có tên Apache Kafka.

Bước 3:

Ứng dụng Cust Hist Import thực hiện đọc bản tin dưới dạng mảng byte trên phân hệ Kafka sau đó lưu vào cơ sở dữ liệu lớn có tên Hbase. Trong quá trình lưu vào Hbase cần chuyển từ đối tượng mảng byte sang CdrEvent theo quy trình mô tả ở trên để lấy thông tin về số điện thoại, loại CDR, thời gian để tạo khóa còn value là mảng byte. Tất cả lưu vào cơ sở dữ liệu lớn Hbase để phục vụ việc tra cứu lịch sử và sử dụng cho các bài toán kinh doanh sau này.

Ứng dụng CBA Accumulate đọc song song với ứng dụng Cust Hist Import để chuyển đổi từ mảng byte sang đối tượng CDREvent. Từ đối lượng này ứng dụng lấy các thông tin tiêu dùng để thực hiện tích lũy tiêu dùng cho thuê bao theo các chu kỳ ngày, tuần, tháng, theo khung giờ, theo gói cước … nhằm mục đích phục vụ các chiến lược kinh doanh, theo dõi doanh thu.

Bước 4:

Ứng dụng Web Cust Hist nhận yêu cầu truy vấn từ chăm sóc khách hàng, hệ thống thực hiện tra cứu lịch sử thuê bao theo số điện thoại, loại dịch vụ, và khoảng thời gian từ cơ sở dữ liệu lớn Hbase. Dữ liệu nhận được dưới dạng mảng byte thực hiện chuyển đổi sang bản tin CDREvent để hiển thị cho người dùng xem thông tin chi tiết cước tiêu dùng.

Ngoài ra các ứng dụng phân tích (CBA Analytics) thực hiện quét cơ sở dữ liệu lớn để tổng hợp dữ liệu tạo các báo cáo, phân tích. Dữ liệu từ mảng byte chuyển đổi sang CDREvent từ đó lấy được các thuộc tính để thực hiện các nghiệp vụ từng báo cáo, phân tích.

Để so sánh thực hiện theo sáng chế, thư viện Protocol Buffer, thư viện Java truyền thống. Xây dựng cấu trúc bản tin gồm:

- 1 thuộc tính kiểu long lưu thời gian tạo: giá trị timestamp 8 byte
- 1 thuộc tính kiểu long lưu thời gian nhận: giá trị timestamp 8 byte
- 1 thuộc tính kiểu string lưu mã dịch vụ: giá trị "1" 1 byte
- 1 thuộc tính kiểu string lưu sessionId: "test@1527237019546"
- 1 thuộc tính map kiểu integer, long: lưu khóa - giá trị

Đánh giá về dung lượng:

So sánh giá trị đưa vào map lần lượt 100 giá trị các kiểu Byte, Short, Integer, Long:

Byte 1:1, 2:2, .. 100:100

Short 1:128, 2:129, .. 100:227

Integer 1:32768, 2: 32769, .. 100:32867

Long 1:2147483648, 2:2147483649 .. 100: 2147483747

Thu được kết quả được thể hiện trong Bảng 9 dưới đây:

Bảng 9: So sánh kích thước bản tin dữ liệu giữa sáng chế, Protocol Buffer, Java với giá trị kiểu số

	Protocol Buffer	Sáng chế	Java
Byte	637	238	2102
Short	737	338	2203
Integer	837	538	2365
Long	1037	938	2802

Thay đổi bản tin kiểu map từ integer, long sang integer, byte[]. Với các giá trị như trên nhưng chuyển từ kiểu số sang mảng byte. Tác giả thu được kết quả được thể hiện trong Bảng 10 dưới đây.

Bảng 10: So sánh kích thước bản tin dữ liệu giữa sáng chế, Protocol Buffer, Java với giá trị kiểu byte array từ số.

	Protocol Buffer	Sáng chế	Java
Byte	737	338	2478
Short	837	438	2578
Int	1037	638	2778
Long	1437	1038	3178

Hình 4 và Hình 5 là các biểu đồ so sánh về kích thước bản tin dữ liệu giữa phương pháp theo sáng chế, Protocol Buffer, Java với giá trị kiểu byte array từ số.

Đánh giá về tốc độ:

Để đánh giá tốc độ, thực hiện serialization trên hệ thống với các thông số cấu hình sau:

CPU: Intel(R) Xeon(R) E5 - 2690 v4 2.6 GHz

RAM: 16 GB

Chạy 1 luồng xử lý.

Đối với bản tin có map chứa 100 thực thể với value là mảng byte[] từ giá trị Long. 1:2147483648, 2:2147483649 .. 100: 2147483747

Thực liện lặp lại 10 lần mỗi lần thực hiện 100.000 thao tác chuyển bản tin sang mảng byte và quá trình ngược lại ta thu được kết quả sau. Số thao tác chuyển đổi trong khoảng đơn vị thời gian giây được gọi là tốc độ (transaction per second viết tắt là TPS)

Bảng 11: Đánh giá tốc độ và thời gian xử lý 100.000 bản tin kiểu mảng byte

	Protocol Buffer		Sáng chế		Java	
STT	TPS	Thời gian	TPS	Thời gian	TPS	Thời gian
1	31766	3148	52192	1916	7805	12812
2	51948	1925	79051	1265	10716	9331
3	61804	1618	83542	1197	10099	9901
4	71942	1390	90497	1105	10707	9339

	Protocol Buffer		Sáng chế		Java	
STT	TPS	Thời gian	TPS	Thời gian	TPS	Thời gian
5	72254	1384	106269	941	10472	9549
6	74460	1343	100704	993	9319	10730
7	73367	1363	57471	1740	9248	10812
8	75131	1331	69060	1448	10000	10000
9	73260	1365	104602	956	10679	9364
10	74128	1349	89686	1115	9967	10033
Avg	66006	1621	83307	1267	9901	10187

Đối với bản tin có map chứa 100 thực thể với value là giá trị Long. 1:2147483648, 2:2147483649 .. 100: 2147483747 ta thu được kết quả:

Bảng 12: Đánh giá tốc độ và thời gian xử lý 100.000 bản tin kiểu số Long

	Protocol Buffer		Sáng chế		Java	
STT	TPS	Thời gian	TPS	Thời gian	TPS	Thời gian
1	48007	2083	44306	2257	7117	14049
2	74515	1342	95785	1044	8238	12138
3	72939	1371	148148	675	7702	12982
4	65832	1519	126742	789	8582	11651
5	68073	1469	151057	662	7928	12613
6	77821	1285	101936	981	7919	12627
7	58582	1707	121951	820	8103	12340
8	72254	1384	149031	671	8259	12108
9	73964	1352	151745	659	8065	12398
10	75585	1323	150602	664	8570	11668
Avg	68757	1483	124130	922	8048	12457

Hình 6 là biểu đồ so sánh tốc độ giữa phương pháp theo sáng chế, Protocol Buffer, Java trên 2 kiểu dữ liệu byte array và số Long.

Đánh giá về tính linh động:

+ Đối với thư viện Java, thực hiện serialization tất cả các kiểu thực thi giao diện Serializable.

+ Đối với Protocol Buffer, thực hiện serialization kiểu dữ liệu đã được định nghĩa trước. Khi thay đổi kiểu dữ liệu thì cần phải tải lên mã mới.

+ Đối với phương pháp theo sáng chế, có thể áp dụng cho các kiểu Byte, Short, Integer, Long, Date, String, byte array.

Hiệu quả đạt được của sáng chế

Phương pháp giảm kích thước bản tin chứa danh sách các ánh xạ khóa - giá trị theo tần suất kiểu dữ liệu khóa - giá trị trong xử lý dữ liệu thời gian thực như sáng chế đã đạt được một số ưu điểm sau:

- Giảm kích thước bản tin so với các thư viện truyền thống Java, Protocol Buffer.

- Đáp ứng yêu cầu xử lý dữ liệu thời gian thực, hiệu năng vượt trội so với phương pháp serialization của thư viện Java cung cấp và cao hơn so với thư viện protocol Buffer.

- Cho phép định nghĩa động giá trị khóa mới.

- Không cần định nghĩa lại, viết lại mã bản tin.

- Kiểu giá trị hỗ trợ đa dạng các kiểu dữ liệu như Byte, Short, Integer, Long, Date, String, byte array.

YÊU CẦU BẢO HỘ

1. Phương pháp giảm kích thước bản tin chứa danh sách các ánh xạ khóa - giá trị theo tần suất kiểu dữ liệu khóa - giá trị trong xử lý dữ liệu thời gian thực, bao gồm các bước sau:

a) xây dựng bảng mã các kiểu dữ liệu cần chuyển đổi của khóa - giá trị:

ai) đánh mã các thuộc tính dữ liệu;

thực hiện đánh mã các thuộc tính ở dạng chuỗi ký tự sang kiểu số nguyên có giới hạn là kiểu Short; các mã này được lưu vào tệp hoặc bảng trong cơ sở dữ liệu;

aii) xây dựng giao diện cho phép đọc các thuộc tính này;

ứng dụng thực hiện chức năng đọc từ tệp hoặc cơ sở dữ liệu; duyệt kết quả đọc tệp hoặc cơ sở dữ liệu thu được bảng mã các thuộc tính;

aiii) định nghĩa kiểu dữ liệu cần chuyển đổi;

định nghĩa kiểu dữ liệu của khóa có 2 loại là Byte và Short;

định nghĩa kiểu dữ liệu của giá trị có 15 loại;

giá trị kiểu Byte có 1 kiểu chuyển đổi B2B;

giá trị kiểu Short có 2 kiểu chuyển đổi S2B và S2S;

giá trị kiểu Integer có 3 kiểu chuyển đổi I2B I2S và I2I;

giá trị kiểu Long có 4 kiểu chuyển đổi L2B L2S L2I và L2L;

giá trị kiểu Date chuyển về kiểu long có 1 kiểu chuyển đổi D2L;

giá trị kiểu String căn cứ theo độ dài có 2 kiểu chuyển đổi BS và SS;

giá trị kiểu Bytearray căn cứ theo độ dài có 2 kiểu chuyển đổi BA và SA;

kết hợp khóa và giá trị định nghĩa ra 30 kiểu dữ liệu chuyển đổi;

định nghĩa này phục vụ cho việc serialization từ bản tin sang mảng byte và ngược lại từ mảng byte sang bản tin;

điểm khác biệt ở chỗ:

sử dụng tối đa 2 byte để định nghĩa kiểu dữ liệu của khóa;

định nghĩa ra 15 loại kiểu dữ liệu chuyển đổi của giá trị;

định nghĩa ra 30 kiểu dữ liệu chuyển đổi của khóa - giá trị;

b) xác định kiểu dữ liệu chuyển đổi cho khóa - giá trị trong bản tin:

bi) xác định kiểu dữ liệu khóa;

khóa là các số nguyên đã được đánh mã từ thuộc tính có giới hạn Short là 2 byte;

với khóa có giá trị nằm trong giới hạn Byte thì xác định là kiểu Byte;

với khóa có giá trị nằm trong giới hạn Short thì xác định kiểu là kiểu Short;

bii) xác định kiểu dữ liệu giá trị;

đối với giá trị ta xác định kiểu dữ liệu tuân theo quy tắc sau:

với giá trị là các số nguyên thì căn cứ theo giá trị thuộc giới hạn kiểu nào ta xác định được kiểu tương ứng;

giá trị kiểu Byte có 1 kiểu chuyển đổi B2B;

giá trị kiểu Short có 2 kiểu chuyển đổi S2B và S2S;

giá trị kiểu Integer có 3 kiểu chuyển đổi I2B I2S và I2I;

giá trị kiểu Long có 4 kiểu chuyển đổi L2B L2S L2I và L2L;

giá trị kiểu Date chuyển về kiểu Long có 1 kiểu chuyển đổi D2L;

giá trị kiểu String căn cứ theo độ dài có 2 kiểu chuyển đổi BS và SS;

giá trị kiểu Bytearray căn cứ theo độ dài có 2 kiểu chuyển đổi BA và SA;

biii) kết hợp khóa và giá trị ta xác định được kiểu khóa - giá trị;

kết hợp kiểu dữ liệu của khóa và kiểu dữ liệu của giá trị ta xác định được kiểu dữ liệu của khóa - giá trị;

với khóa xác định được 1 trong 2 kiểu là Byte (B) hoặc Short (S);

với giá trị xác định được 1 trong 15 kiểu B2B S2B S2S I2B I2S I2I L2B L2S L2I L2L D2L BS SS BA SA;

kết hợp kiểu khóa và kiểu giá trị xác định được kiểu khóa - giá trị là 1 trong 30 kiểu BB2B BS2B BS2S BI2B BI2S BI2I BL2B BL2S BL2I BL2L BD2L BBS BSS BBA BSA SB2B SS2B SS2S SI2B SI2S SI2I SL2B SL2S SL2I SL2L SD2L SBS SSS SBA SSA;

điểm khác biệt ở chỗ:

xác định kiểu dữ liệu khóa theo định nghĩa 2 kiểu dữ liệu của khóa;

xác định kiểu dữ liệu giá trị theo định nghĩa 15 kiểu dữ liệu chuyển đổi của giá trị;

xác định kiểu khóa - giá trị theo định nghĩa 30 loại kiểu dữ liệu của khóa - giá trị;

c) gom các cặp khóa - giá trị có cùng kiểu định nghĩa vào một nhóm:

ci) duyệt tất cả các cặp khóa - giá trị;

cii) gom các cặp khóa - giá trị cùng kiểu định nghĩa vào 1 nhóm;

điểm khác biệt ở chỗ gom các cặp khóa - giá trị cùng kiểu định nghĩa vào 1 nhóm;

d) lưu các cặp khóa - giá trị đã được chuyển đổi theo nhóm vào vùng nhớ đệm:

di) ghi kiểu dữ liệu cặp khóa - giá trị của nhóm;

dii) ghi số lượng các cặp có cùng kiểu;

diii) ghi dữ liệu các cặp khóa - giá trị trong nhóm;

duyệt các cặp khóa - giá trị trong nhóm thực hiện;

ghi dữ liệu của khóa;

với kiểu giá trị không phải bytearray hoặc string thì ghi tiếp giá trị;

với kiểu giá trị là bytearray hoặc string thì ghi độ dài mảng byte sau đó là ghi giá trị;

điểm khác biệt ở việc lưu thông tin gồm ghi kiểu dữ liệu cặp khóa - giá trị của nhóm, ghi số lượng các cặp có cùng kiểu, ghi dữ liệu các cặp khóa - giá trị trong nhóm;

e) sao chép từ vùng nhớ đệm vào mảng byte kết thúc việc serialization từ đối tượng sang mảng byte; và

f) thực hiện chuyển mảng byte sang kiểu bản tin bằng cách xác định kiểu dữ liệu dựa vào giá trị của byte:

fi) xác định kiểu dữ liệu của nhóm;

đọc 1 byte để xác định kiểu dữ liệu của nhóm;

fii) xác định số lượng cặp trong nhóm;

đọc 1 byte tiếp theo kiểm tra bit đầu tiên của byte; nếu là 0 thì số lượng cặp chỉ ở giới hạn byte; nếu là 1 thì đọc tiếp 1 byte tiếp ta tính ra số lượng cặp khóa - giá trị của kiểu này;

fiii) đọc các cặp khóa - giá trị trong nhóm cho đủ số lượng cặp; căn cứ vào kiểu dữ liệu của nhóm để đọc cặp khóa - giá trị;

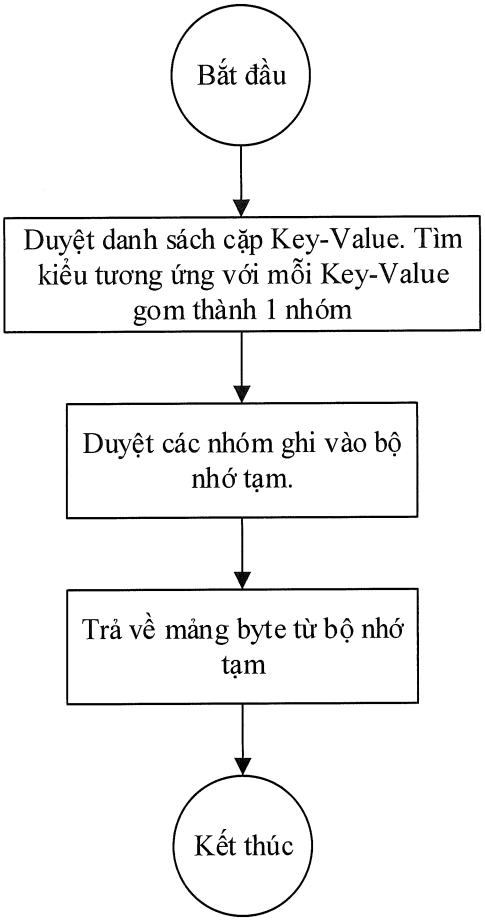
nếu kiểu nhóm có kiểu khóa thuộc loại 1 byte thì đọc 1 byte để xác định khóa; nếu kiểu nhóm có kiểu khóa thuộc loại 2 byte thì đọc 2 byte để xác định

khóa; đối với giá trị căn cứ theo kiểu nhóm để xác định kiểu dữ liệu của giá trị và số byte giá trị chiếm để đọc; với string và bytearray thì cần xác định độ dài X của mảng byte theo kiểu của giá trị để đọc ra được độ dài; sau đó đọc tiếp X byte để thu được giá trị của dạng bytearray của 2 kiểu này;

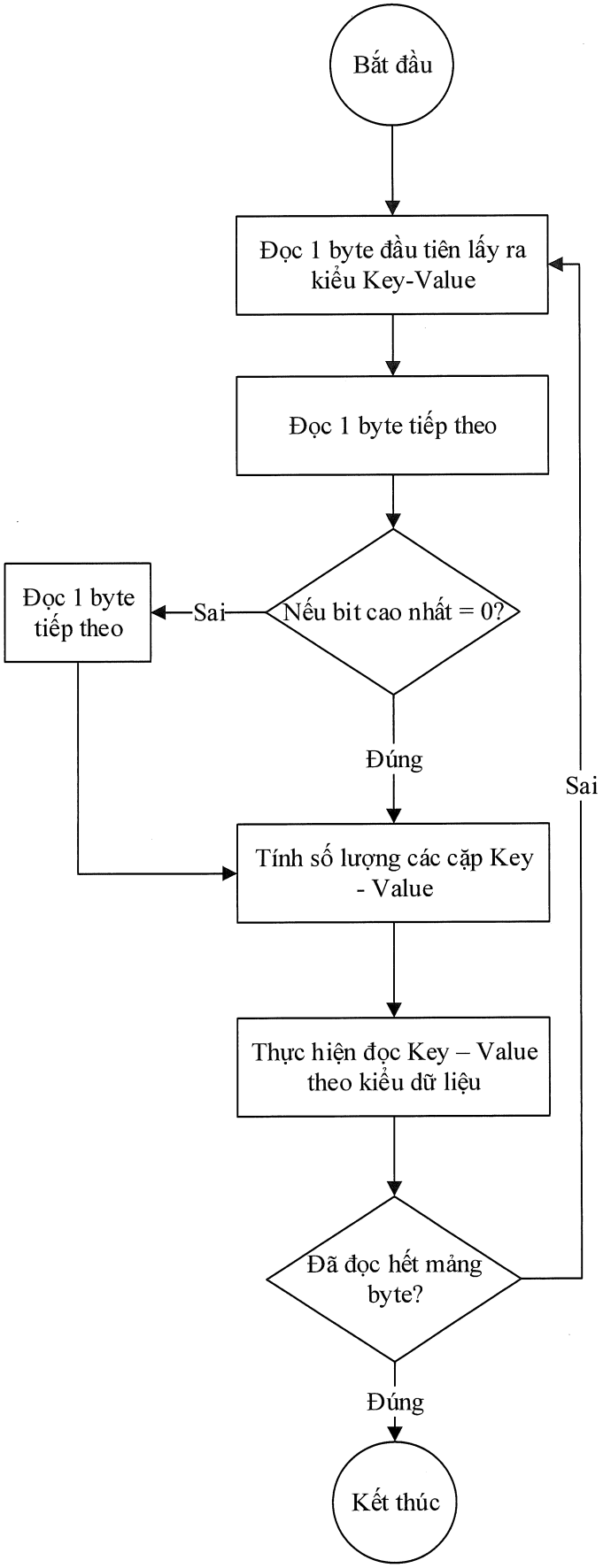
fiv) đọc lặp lại cho đến hết mảng byte;

điểm khác biệt ở chỗ:

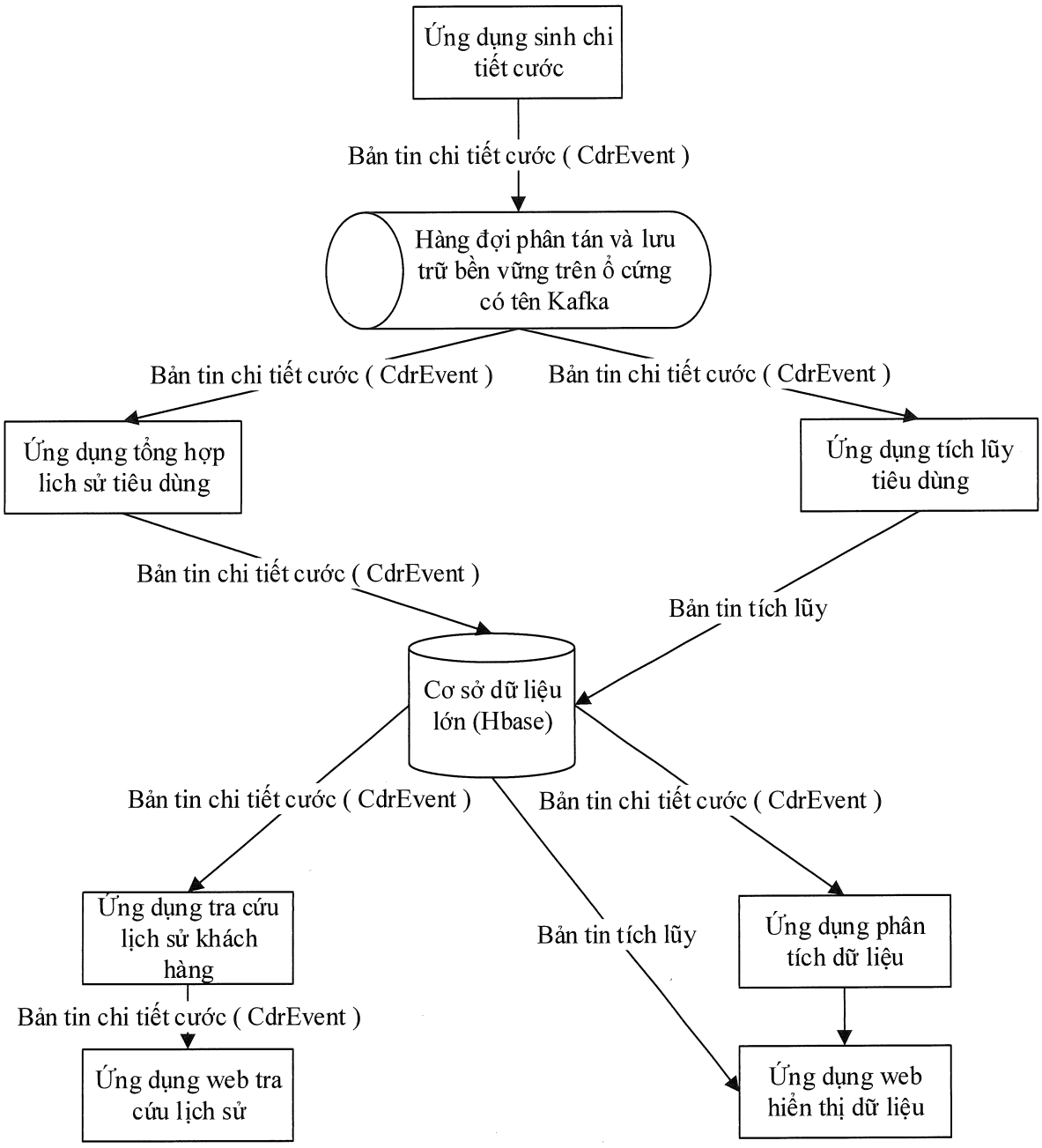
xác định kiểu dữ liệu của nhóm theo 30 kiểu đã định nghĩa; trình tự đọc kiểu dữ liệu của nhóm, số lượng cặp trong nhóm, cặp khóa - giá trị trong nhóm căn cứ theo kiểu dữ liệu của nhóm.


Bắt đầu
Duyệt danh sách cặp Key-Value. Tìm kiểu tương ứng với mỗi Key-Value gom thành 1 nhóm
Duyệt các nhóm ghi vào bộ nhớ tạm.
Trả về mảng byte từ bộ nhớ tạm
Kết thúc

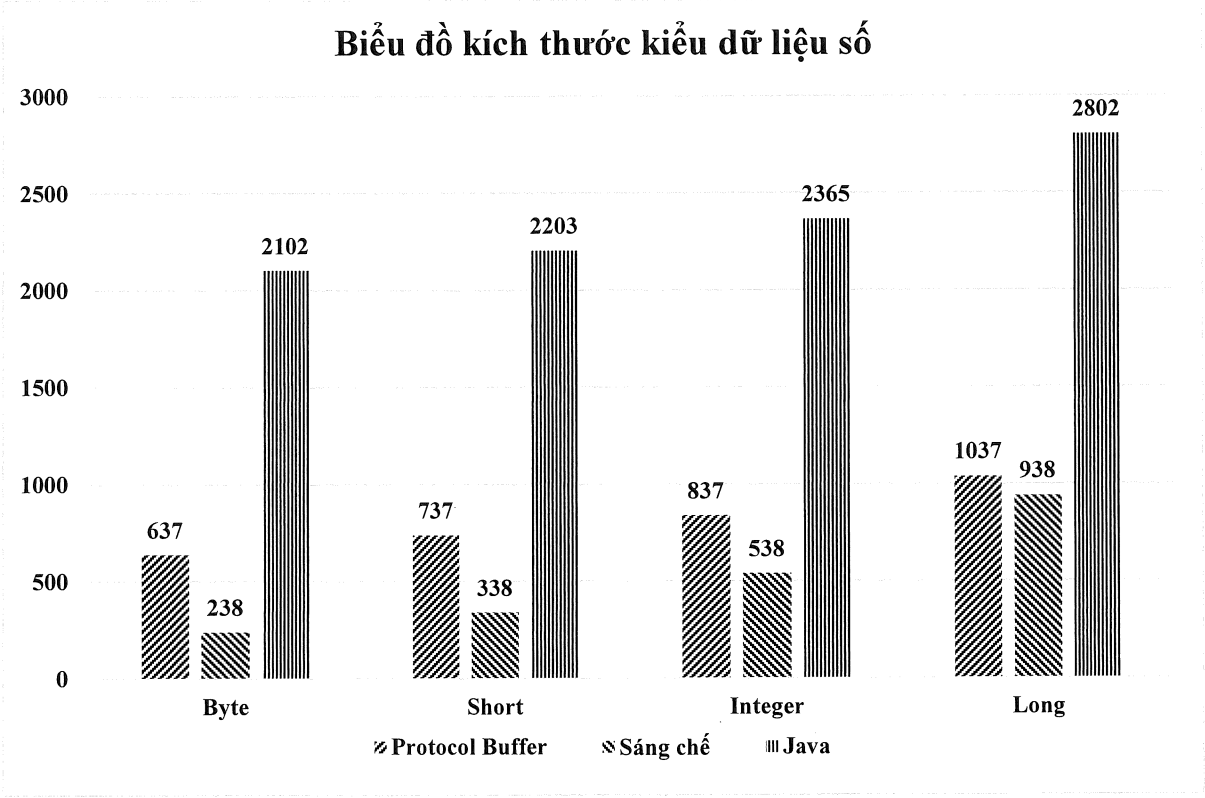
Hình 1


Bắt đầu
Đọc 1 byte đầu tiên lấy ra kiểu Key-Value
Đọc 1 byte tiếp theo
Nếu bit cao nhất = 0?
Sai
Đọc 1 byte tiếp theo
Đúng
Sai
Tính số lượng các cặp Key - Value
Thực hiện đọc Key – Value theo kiểu dữ liệu
Đã đọc hết mảng byte?
Đúng
Kết thúc

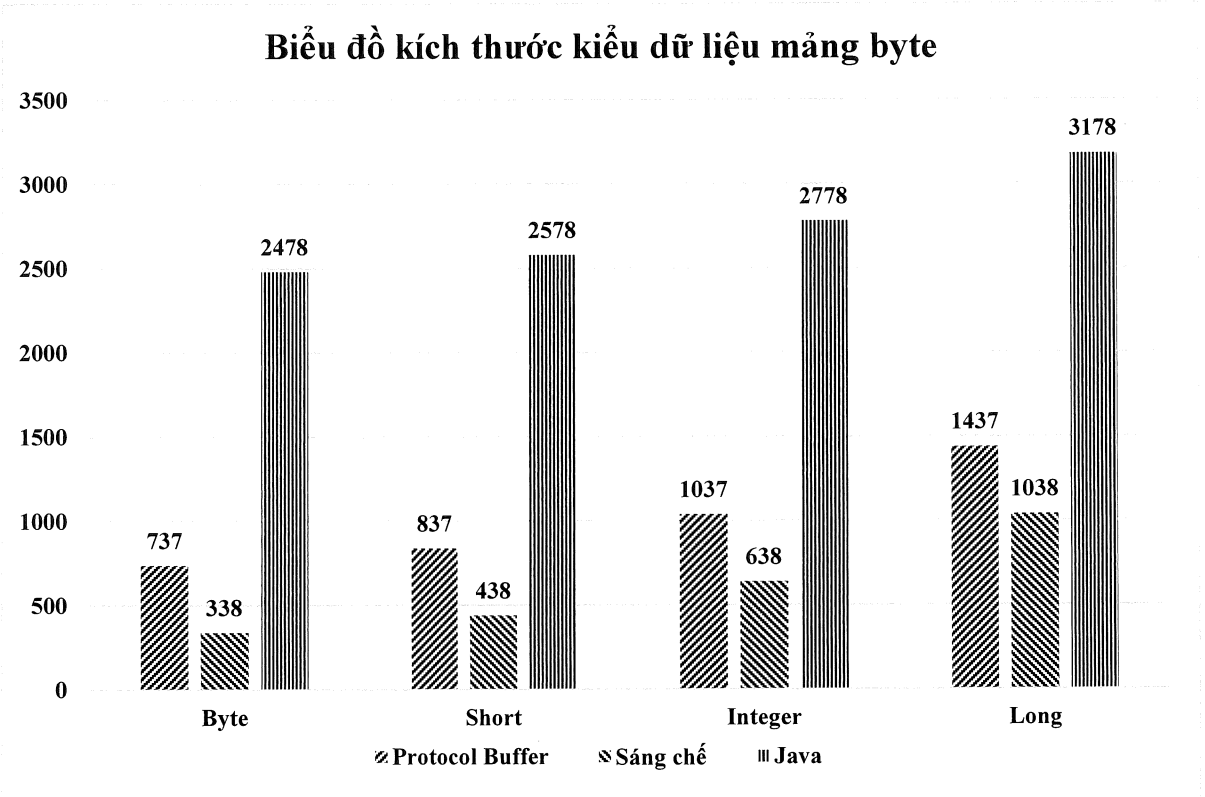
Hình 2


Ứng dụng sinh chi tiết cước
Bản tin chi tiết cước (CdrEvent)
Hàng đợi phân tán và lưu trữ bền vững trên ổ cứng có tên Kafka
Bản tin chi tiết cước (CdrEvent)
Bản tin chi tiết cước (CdrEvent)
Ứng dụng tổng hợp lich sử tiêu dùng
Ứng dụng tích lũy tiêu dùng
Bản tin chi tiết cước (CdrEvent)
Bản tin tích lũy
Cơ sở dữ liệu lớn (Hbase)
Bản tin chi tiết cước (CdrEvent)
Bản tin chi tiết cước (CdrEvent)
Ứng dụng tra cứu lịch sử khách hàng
Bản tin tích lũy
Ứng dụng phân tích dữ liệu
Bản tin chi tiết cước (CdrEvent)
Ứng dụng web tra cứu lịch sử
Ứng dụng web hiển thị dữ liệu

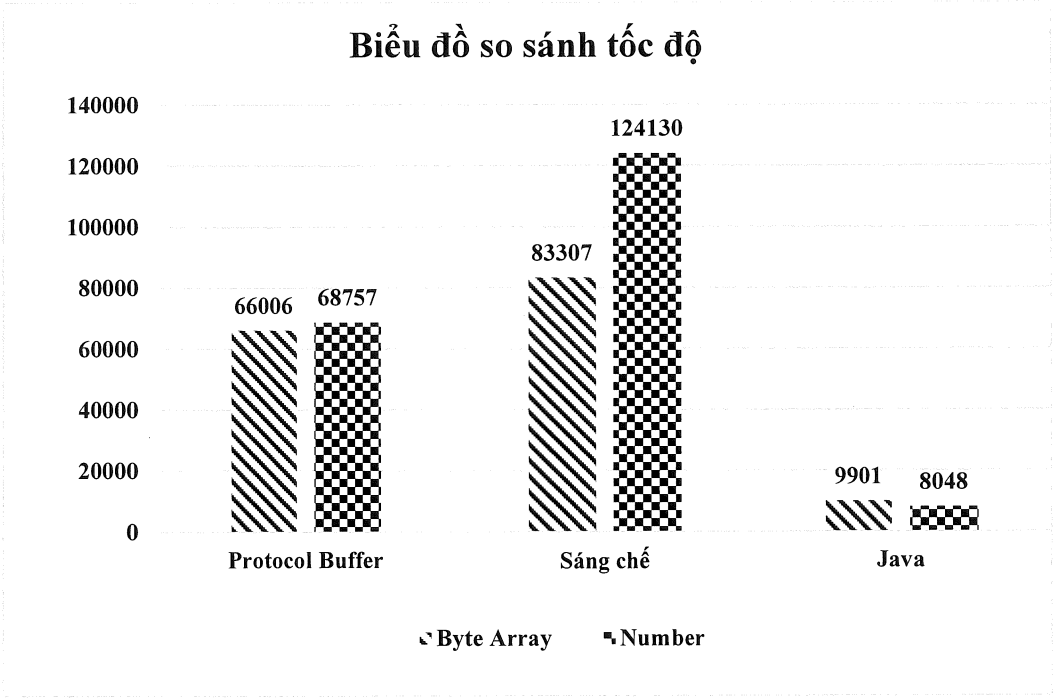
Hình 3


Biểu đồ kích thước kiểu dữ liệu số
3000
2500
2000
1500
1000
500
0
637
238
2102
737
338
2203
837
538
2365
1037
938
2802
Byte
Short
Integer
Long
Protocol Buffer
Sáng chế
Java

Hình 4


Biểu đồ kích thước kiểu dữ liệu mảng byte
3500
3000
2500
2000
1500
1000
500
0
737
338
2478
837
438
2578
1037
638
2778
1437
1038
3178
Byte
Short
Integer
Long
Protocol Buffer
Sáng chế
Java

Hình 5


Biểu đồ so sánh tốc độ
140000
120000
100000
80000
60000
40000
20000
0
66006
68757
83307
124130
9901
8048
Protocol Buffer
Sáng chế
Java
Byte Array
Number

Hình 6