ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
Khoa Công Nghệ Thông Tin

Khai thác chuỗi tuần tự từ cơ sở dữ liệu chuỗi trên Hadoop MapReduce

CBHD: PGS. TS. Lê Hoài Bắc

HVTH: HOÀNG ĐỨC THỌ

TP. Hồ Chí Minh, 09/2017

Mục tiêu

- Nghiên cứu các cách tiếp cận trong khai thác dữ liệu chuỗi, cùng với việc tổ chức cấu trúc dữ liệu và các kỹ thuật tối ưu trong quá trình khai thác.

- Áp dụng và đề xuất cải tiến kỹ thuật khai thác nhằm nâng cao hơn nữa hiệu suất của quá trình khai thác. Kết hợp với kỹ thuật lập trình song song và chia nhỏ dữ liệu trên mô hình Hadoop MapReduce.

Nội dung trình bày

1. Giới thiệu
2. Khai thác chuỗi
3. Phương pháp đề xuất
4. Kết quả thực nghiệm
5. Kết luận và hướng phát triển

1. Giới thiệu

Lĩnh vực ứng dụng

- Phân tích thói quen mua sắm của khách hàng
- Hành vi sử dụng Web
- Xử lý văn bản
- …

Khai thác cơ sở dữ liệu (CSDL) chuỗi thường được chia làm hai giai đoạn chính:

1) Giai đoạn khai thác mẫu tuần tự
2) Giai đoạn sinh luật từ các mẫu của Giai đoạn đầu

1. Giới thiệu (tt)

Khai thác mẫu tuần tự: tìm các chuỗi con phổ biến thỏa điều kiện ngưỡng hỗ trợ (minSup) cho trước từ CSDL chuỗi

SID	Sequences
1	({A},{A,B},{A,B,C})
2	({A,B},{A,C},{B,C})
3	({C},{A,C},{A,B})
4	({D},{A,B,D})

SDB

minSup = 50%

Patterns	Supports
({A})	4
({A},{A})	3
({A},{B})	3
({A},{C})	2
({A},{A},{B})	2
…	…

Một số mẫu tuần tự

Thách thức

- Không gian tìm kiếm rộng
- Kích thước tập dữ liệu ngày càng tăng

2. Khai thác chuỗi
A. Kỹ thuật khai thác tuần tự

- Phân loại theo đặc trưng biểu diễn dữ liệu
  - Kỹ thuật định dạng dữ liệu dạng ngang: **AprioriAll**, **GSP**
  - Kỹ thuật định dạng dữ liệu dạng dọc: **SPAM**, **SPADE**, **LAPIN-SPAM**
  - Kỹ thuật phát triển mẫu: **FreeSpan**, **PrefixSpan**

- Các thuật toán biểu diễn dữ liệu theo dạng dọc hiệu quả hơn
  - Chỉ duyệt CSDL 1 hoặc 2 lần
  - Khai thác trên bảng dữ liệu đã chuyển đổi
  - Tính độ hỗ trợ bằng các phép giao logic, phép xử lý bit AND

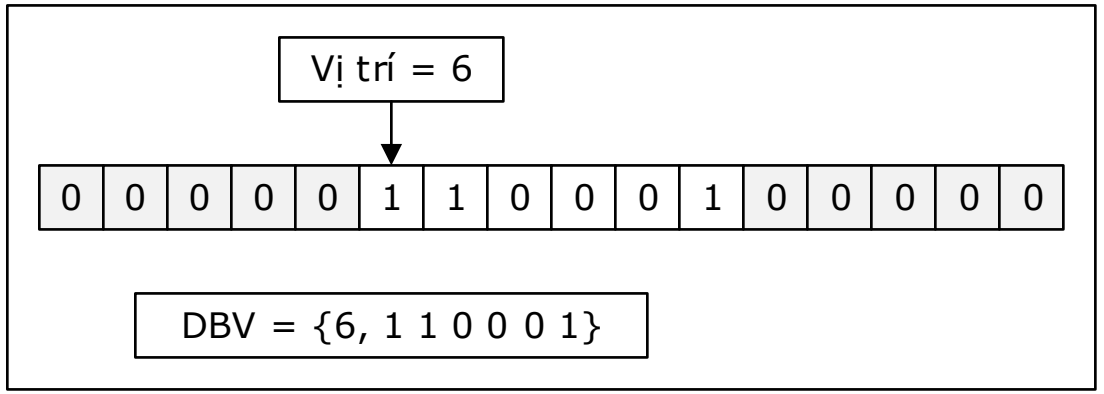
2. Khai thác chuỗi
A. Kỹ thuật khai thác tuần tự (tt)

- **SPADE**: cấu trúc IDList
- **SPAM**, **bitSPADE**, **LAPIN-SPAM**: cấu trúc vector bit
  - Giá trị 1 tương ứng với sự xuất hiện của sự kiện trong giao dịch
  - Ngược lại, gán giá trị 0

- Hạn chế
  - Phát sinh nhiều mẫu ứng viên dư thừa
  - Sử dụng vector bit có kích thước cố định

2. Khai thác chuỗi
A. Kỹ thuật khai thác tuần tự (tt)

- Cấu trúc vector bit động (Dynamic Bit Vector - DBV)
  - Được đề xuất bởi Võ và cộng sự trong thuật toán DBV-Miner (2012)
  - Loại bỏ các giá trị 0 ở đầu và cuối vector bit
  - Gồm 2 thành phần:
    - Vị trí bắt đầu (start)
    - Cấu trúc vector bit (vector bit)



2. Khai thác chuỗi
A. Kỹ thuật khai thác tuần tự (tt)

- Cấu trúc ánh xạ đồng xuất hiện (Co-occurrence MAP - CMAP)
  - Philippe Fournier-Viger đề xuất năm 2014
  - Tập các sự kiện có thể mở rộng của các sự kiện phổ biến
  - Được xem như bảng tham chiếu từ các mẫu tuần tự có độ dài 2 (2-sequence)

Sự kiện	Tập mở rộng trong S-Step	Tập mở rộng trong I-Step	Tập 2-sequence
A	{A, B, C}	{B,C}	({A},{A}),({A},{B}),({A},{C}) ({A,B}),({A,C})
B	{A, B, C}	{C}	({B},{A}),({B},{B}),({B},{C}) ({B,C})
C	{B, C}		({C},{B}),({C},{C})

Cấu trúc CMAP của SDB với minSup = 50%

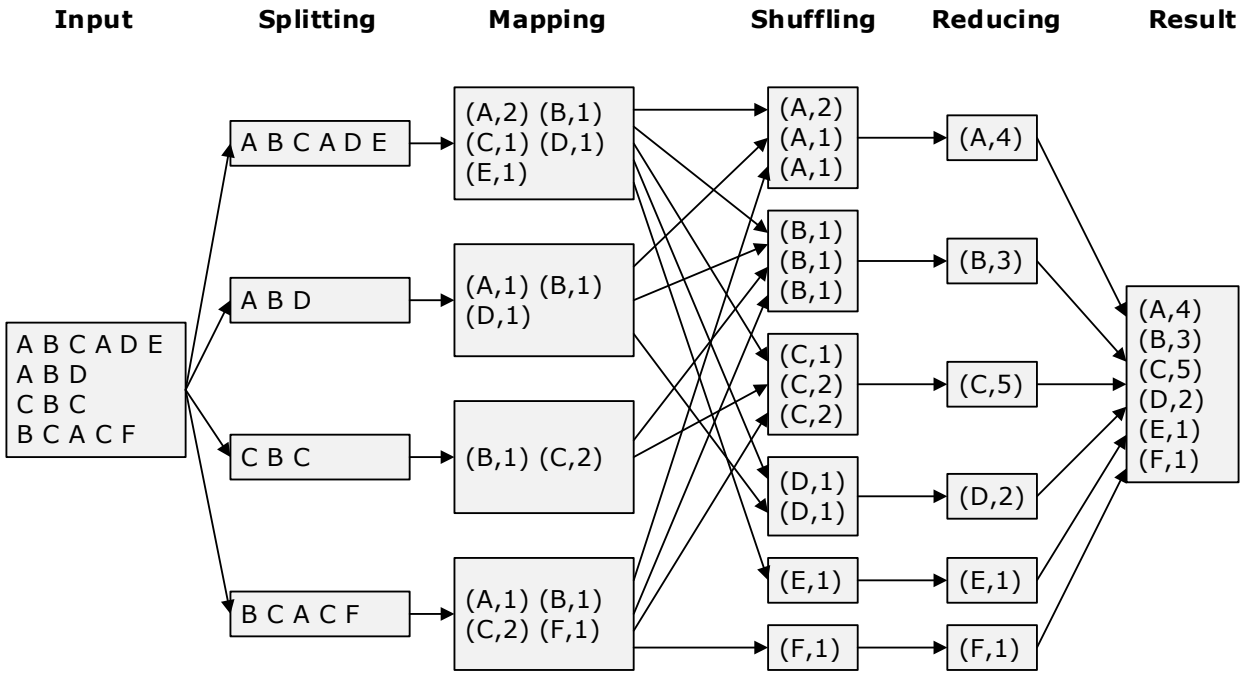
2. Khai thác chuỗi
B. Kỹ thuật khai thác song song

- Khai thác song song: Tăng hiệu quả khai thác và khả năng mở rộng
  - **pSPADE**: dựa trên hệ thống chia sẻ bộ nhớ (share-memory based)
  - **tree-projection-based**: dựa trên hệ thống phân tán bộ nhớ (distributed-memory based)
  - **PIB-PRISM**, **pDBV-SPM** : dựa trên vi xử lý đa lõi (multi-core processor)

- Hạn chế:
  - Thực hiện trên máy đơn
  - Giới hạn tài nguyên
  - Chi phí giao tiếp
  - Khai thác không cân bằng tải (load unbalancing)

2. Khai thác chuỗi
C. Kỹ thuật khai thác phân tán

- Áp dụng mô hình lập trình phân tán Hadoop MapReduce
  - Thực thi song song
  - Xử lý dữ liệu lớn
  - Dễ mở rộng



2. Khai thác chuỗi
C. Kỹ thuật khai thác phân tán (tt)

- **PTDS**
  - Thực hiện 3 tiến trình MapReduce:
    - Sắp xếp chuỗi
    - Kết hợp các chuỗi cùng tập sự kiện đầu tiên
    - Khai thác mẫu tuần tự: Thực hiện dựa trên ý tưởng thuật toán PrefixSpan
  - Khuyết điểm
    - Khai thác tất cả tập mẫu ứng viên có thể có => Không hiệu quả khi kích thước chuỗi dài

2. Khai thác chuỗi
C. Kỹ thuật khai thác phân tán (tt)

- **SPAMC**
  - Thực hiện dựa trên ý tưởng thuật toán SPAM
  - Khai thác mẫu ứng viên trên các cây con cùng cấp (depth = 2)
    - Loại bỏ sớm mẫu dư thừa
    - Cân bằng tải giữa các hàm Mapper
  - Gọi lặp lại tiến trình MapReduce: kết thúc khi không có mẫu tuần tự mới
  - Khuyết điểm
    - Không hiệu quả khi CSDL chuỗi có số lượng sự kiện phân biệt lớn
    - Phát sinh mẫu ứng viên dư thừa vẫn còn nhiều
    - Chi phí chuyển dữ liệu giữa hàm Mapper và Reducer cao

3. Phương pháp đề xuất
A. Biểu diễn dữ liệu

- Cấu trúc DBVItem: biểu diễn thông tin vị trí xuất hiện của mỗi sự kiện
  - DBVItem <Item, BlockInfo>
  - BlockInfo (DBV, PostList)

SID	Sequences
1	({A},{A,B},{A,B,C})
2	({A,B},{A,C},{B,C})
3	({C},{A,C},{A,B})
4	({D},{A,B,D})

→

Item	({A})	({B})	({C})	({D})
BlockInfo				
DBV	{1:1111}	{1:1111}	{1:111}	{4:1}
PostList	({1,2,3},{1,2},{2,3},{2})	({2,3},{1,3},{3},{2})	({3},{2,3},{1,2})	({1,2})

Chuyển đổi SDB sang cấu trúc DBVItem

- Cấu trúc DBVPattern: biểu diễn thông tin vị trí xuất hiện của chuỗi
  - DBVPattern <Pattern, BlockInfo>

3. Phương pháp đề xuất
A. Biểu diễn dữ liệu (tt)

- Phát sinh mẫu ứng viên: mở rộng mẫu phổ biến DBVPattern từ các sự kiện phổ biến DBVItem bằng 2 hình thức
  - Mở rộng chuỗi (sequence extension)

Pattern or Item	DBV Start Bit	DBV Bit Vector	PostList 1	2	3	4
({A})	1	1111	{1,2,3}	{1,2}	{2,3}	{2}
({A})	1	1111	{1,2,3}	{1,2}	{2,3}	{2}
({A},{A})	1	111	{2,3}	{2}	{3}	

  - Mở rộng tập sự kiện (itemset extension)

Pattern or Item	DBV Start Bit	DBV Bit Vector	PostList 1	2	3	4
({A})	1	1111	{1,2,3}	{1,2}	{2,3}	{2}
({B})	1	1111	{2,3}	{1,3}	{3}	{2}
({A,B})	1	1111	{2,3}	{1}	{3}	{2}

- Cấu trúc DBVTree: lưu trữ các mẫu ứng viên
  - Mỗi nút trên cây là một DBVPattern

3. Phương pháp đề xuất
B. Thuật toán DSPDBV (tt)

- Cho trước CSDL chuỗi SDB và chuỗi S (S ∈ D)
- Giả sử SDB được chia thành n tập dữ liệu (partition), SDB = $p_1 + p_2 + \ldots + p_n$.
- Ta có, $sup(S/SDB) = sup(S/p_1) + sup(S/p_2) + \ldots + sup(S/p_n)$

Nhận xét 1: Chuỗi S được gọi là phổ biến nếu

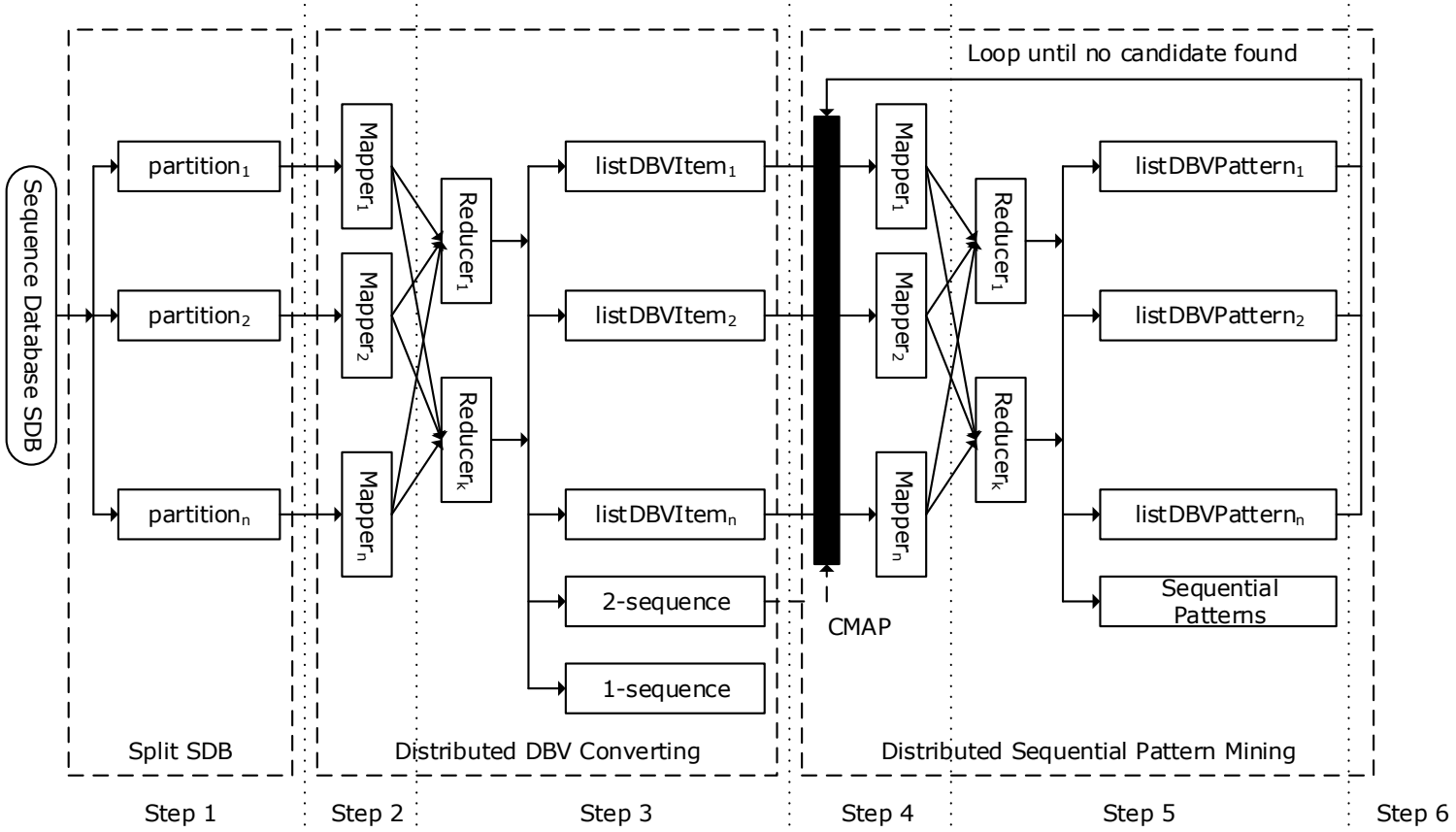
$$sup(S/p_1) + sup(S/p_2) + \ldots + sup(S/p_n) >= minSup.$$

Nhận xét 2: Giả sử chuỗi S là mẫu phổ biến bất kỳ của SDB, nếu S tồn tại trong p_i ($1 \le i \le n$) thì S cũng được xem là mẫu phổ biến trên p_i.

Nhận xét 3: Cho 2 chuỗi S_A và S_B, trong đó $|S_A| < |S_B|$. Với ngưỡng hỗ trợ đủ nhỏ, thời gian phát sinh tập mẫu ứng viên trong S_A sẽ ít hơn đối với S_B do S_A có ít mẫu ứng viên hơn S_B.

3. Phương pháp đề xuất
B. Thuật toán DSPDBV (tt)

- Quá trình khai thác của DSPDBV



3. Phương pháp đề xuất
B. Thuật toán DSPDBV (tt)

- Khai thác phân tán sử dụng MapReduce
  - Hàm Mapper: thực hiện quá trình chuyển đổi dữ liệu, phát sinh mẫu ứng viên.
    - Bước 2 và 4 trong DSPDBV
  - Hàm Reducer: thực hiện tính tổng độ hỗ trợ của mẫu ứng viên.
    - Bước 3 và 5 trong DSPDBV

Algorithm 1: DSPDBV

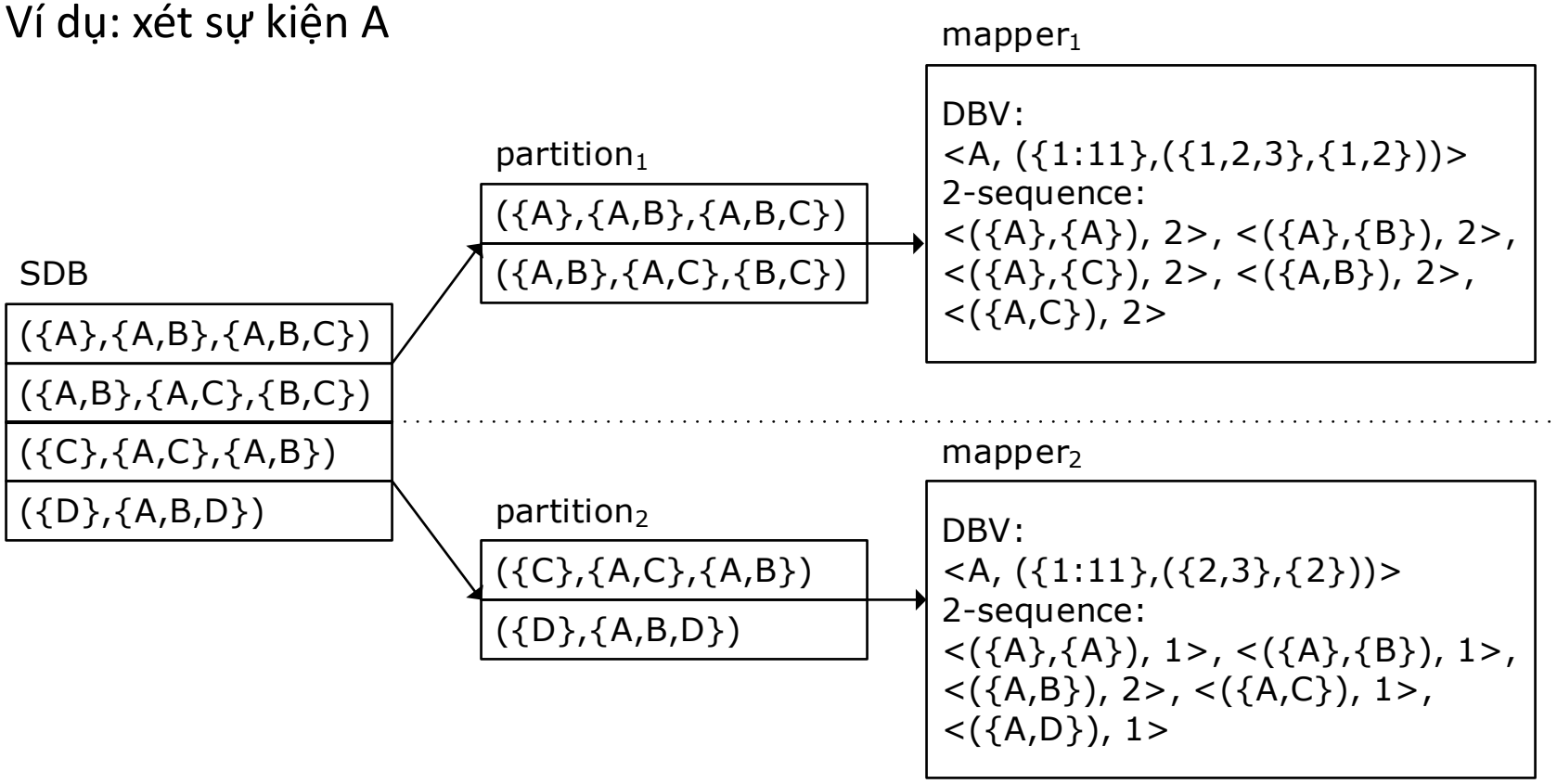
Input: A sequence dataset*SDB*, a support threshold *minSup*,and the maximum depth of sub-prefix tree *depth*

Output: Complete set of frequent sequential patterns

```
1.    Let partitions = split SDB into n partitions;
2.    Call DistributedDBVConversion (partitions, minSup);
3.    While (listDBVPatterns != null)
4.       Call DistributedSequentialPatternMining (listDBVPattens, minSup, depth);
```

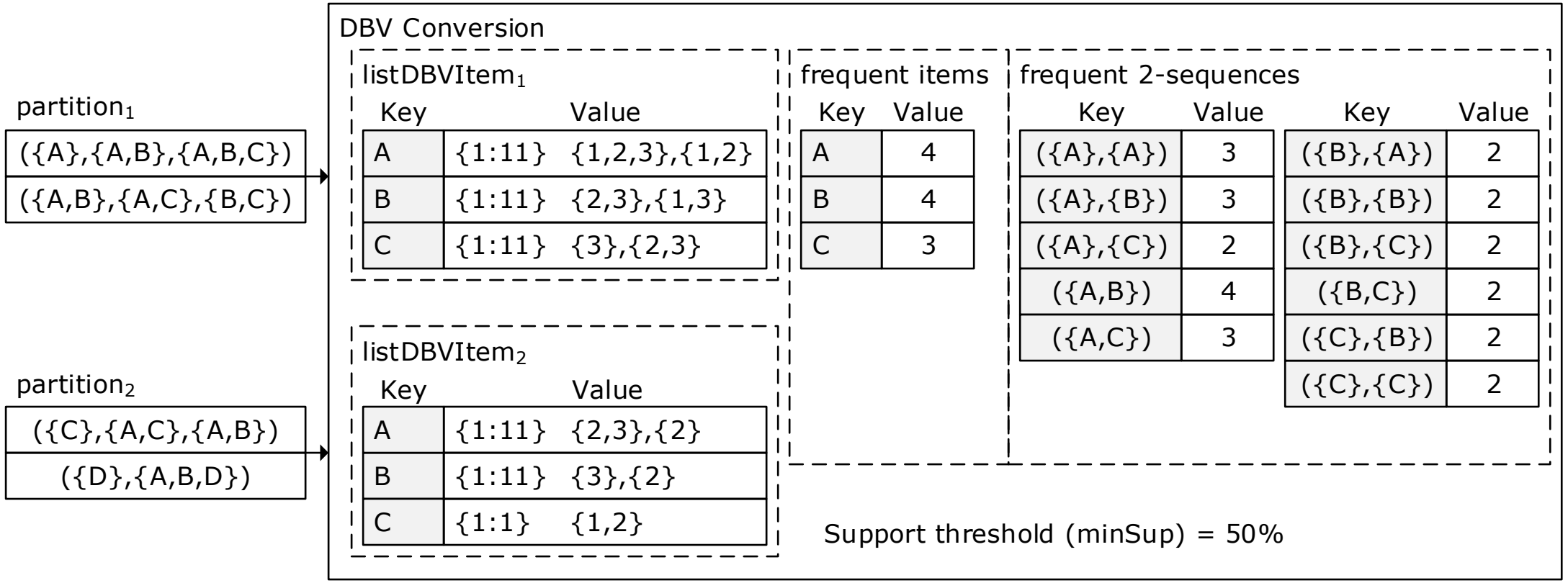
3. Phương pháp đề xuất
B. Thuật toán DSPDBV (tt)

- Phân tán quá trình chuyển đổi dữ liệu
  - Ví dụ: xét sự kiện A



3. Phương pháp đề xuất
B. Thuật toán DSPDBV (tt)

- Phân tán quá trình chuyển đổi dữ liệu (tt)



Kết quả chuyển đổi SDB với minSup = 50%

3. Phương pháp đề xuất
B. Thuật toán DSPDBV (tt)

- Phân tán quá trình chuyển đổi dữ liệu (tt)

Algorithm 2: DistributedDBVConversion

Mapper Side

Input: A partition of sequence dataset *p*

1. Scan partition data only one time to
   a. Construct DBVItem for each item, put into DBVItems<item, blockInfo>
   b. Find 2-sequences and its support, put into sequences<pattern, support>
2. Output <dbv.item, dbv.blockInfo> in DBVItems
3. Output <item.pattern, item.support> in sequences

Reducer Side

Input: Mapper output pairs <*key, values*> and a support threshold *minSup*

Output: Complete set of frequent items, 2-sequences and set of DBVItems divide by partition (listDBVItems)

```
4.  if (key is 2-sequence)
5.      let support = sum of sup(value) in values;
6.      if (support >= minSup)
7.          output <key, support>;
8.  else if (key is item)
9.      let support = sum of sup(value) in values;
10.     add value to blockInfos;
11.     if (support >= minSup)
12.         output <key, support>;
13.         output <key, blockInfo> in blockInfos of each partition;
```

3. Phương pháp đề xuất
B. Thuật toán DSPDBV (tt)

- Phân tán quá trình khai thác mẫu tuần tự

Algorithm 3: DistributedSequentialPatternMining

Mapper Side

Input: A list of root nodes of sub-prefix tree *roots*, a support threshold *minSup,* and the maximum depth of sub-prefix tree *depth*

1. Scan listDBVItem of a partition only one time to
   a. Load list of DBVItems info DBV<item, blockInfo>
   b. Extract frequent list items into items<item>
2. Scan 2-sequence data only one time to construct CMAP
3. Pattern extension for each root in roots

Reducer Side

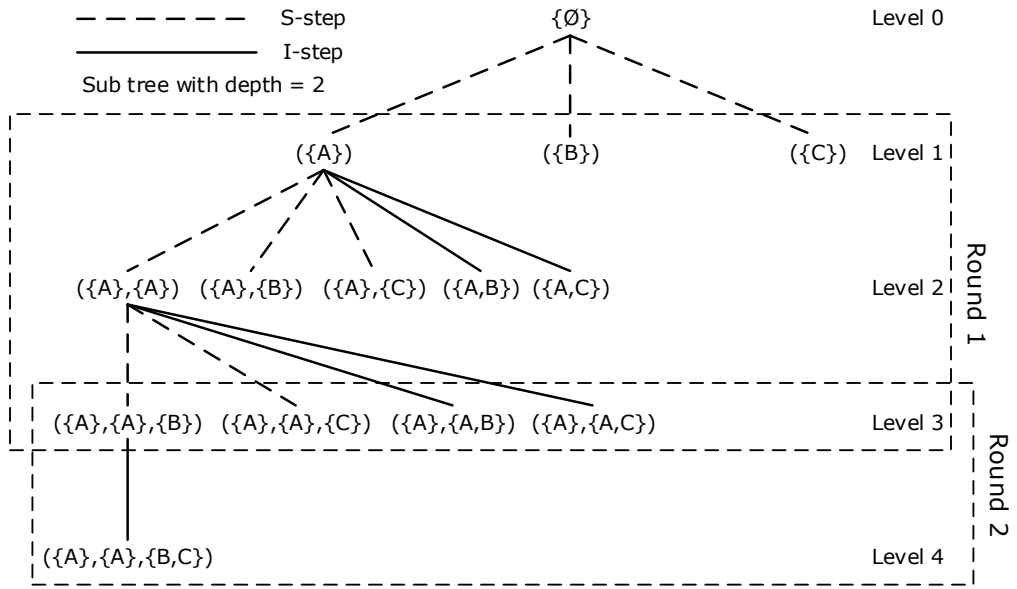
Input: Mapper output pairs <*key, values*>, a support threshold *minSup,* and the maximum depth of sub prefix tree *depth*

Output: Complete set of frequent sequential patterns and set of DBVPatterns divide by partition (listDBVPatterns)

4. **for each** (value in values)
5. let support = sum of sup(value) in values;
6. **if** (key is node at lowest depth in sub-prefix tree)
7. add value to blockInfos
8. **if** (support >= minSup)
9. output <key, support>;
10. **if** (key is node at lowest depth in sub-prefix tree)
11. output <key, blockInfo> in blockInfos in each partition;

3. Phương pháp đề xuất
B. Thuật toán DSPDBV (tt)

- Phân tán quá trình khai thác mẫu tuần tự (tt)
  - Khai thác trên các cây con cùng cấp



Quá trình khai thác trên cây con trong DSPDBV

Algorithm 4: DBVPattenExtension

Input: a root node *root*, a set of frequent items *DBV*, a co-occurrence map *CMAP*, list of extendable items *sItems*, *iItems* and depth information *depth*, *maxDepth*

```
1.  if (depth > maxDepth)
2.      return;
3.  let sNode = sequence-extension for each item in sItem // item is not pruned by CMAP-sequence-extension
4.  for each (node in sNode)
5.      call OutputDBVPatten(node, depth == maxDepth)
6.      Pattern extension for node
7.  let iNode = itemset-extension for each item in iItem // item is not pruned by CMAP-itemset-extension
8.  for each (node in iNode)
9.      call OutputDBVPatten(node, depth == maxDepth)
10.     Pattern extension for node
```

Algorithm 5: OutputDBVPatten

Input: a node of prefix tree *node* and flag to check if node is at lowest depth *flag*

```
1.  if (flag)
2.      output <node.pattern, node.BlockInfo>;
3.  Else
4.      output <node.pattern, sup(node)>;
```

4. Kết quả thực nghiệm
A. Tập dữ liệu

- Bộ dữ liệu thực (http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php)
  - Kiểm tra tính chính xác

Bộ dữ liệu	Số chuỗi	Số sự kiện phân biệt	Loại dữ liệu
BMSWebView1	59601	497	web click stream
BMSWebView2	77512	3340	web click stream
Kosarak10k	10000	10094	web click stream

- Bộ dữ liệu tổng hợp (được phát sinh từ công cụ được cung cấp bởi IBM)
  - Đánh giá: thời gian thực thi và tính mở rộng

Tham số	Ý nghĩa
D	Số lượng chuỗi
C	Số lượng giao dịch trung bình trong chuỗi
T	Số lượng sự kiện trung bình trong tập sự kiện
N	Số lượng sự kiện phân biệt có trong CSDL

4. Kết quả thực nghiệm
A. Tập dữ liệu

- Bộ dữ liệu thực (http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php)
  - Kiểm tra tính chính xác

Bộ dữ liệu	Số chuỗi	Số sự kiện phân biệt	Loại dữ liệu
BMSWebView1	59601	497	web click stream
BMSWebView2	77512	3340	web click stream
Kosarak10k	10000	10094	web click stream

- Bộ dữ liệu tổng hợp (được phát sinh từ công cụ được cung cấp bởi IBM)
  - Đánh giá: thời gian thực thi và tính mở rộng

Tham số	Ý nghĩa
D	Số lượng chuỗi
C	Số lượng giao dịch trung bình trong chuỗi
T	Số lượng sự kiện trung bình trong tập sự kiện
N	Số lượng sự kiện phân biệt có trong CSDL

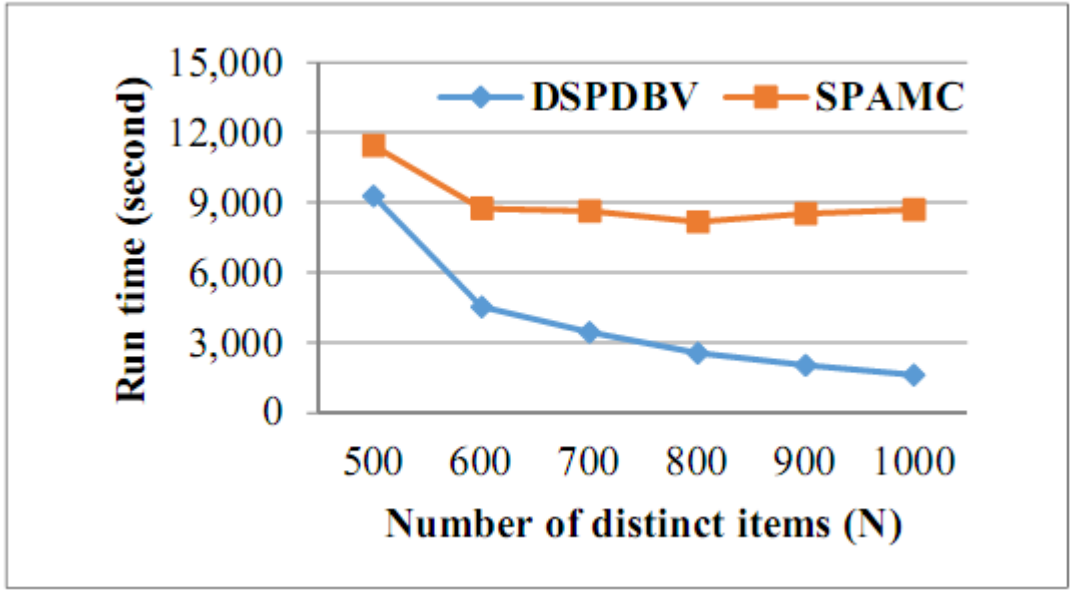
4. Kết quả thực nghiệm
B. Kiểm tra tính chính xác

Bộ dữ liệu	Ngưỡng hỗ trợ (%)	Số lượng mẫu tuần tự khai thác được			
		PrefixSpan	SPAM	SPAMC	DSPDBV
BMSWebView1	0.4	286	286	286	286
	0.5	201	201	201	201
	0.6	162	162	162	162
	0.7	133	133	133	133
BMSWebView2	0.4	676	676	676	676
	0.5	408	408	408	408
	0.6	257	257	257	257
	0.7	187	187	187	187
Kosarak10k	0.4	2800	2800	2800	2800
	0.5	1716	1716	1716	1716
	0.6	1178	1178	1178	1178
	0.7	825	825	825	825

DSPDBV có kết quả tương đồng với các thuật toán: PrefixSpan, SPAM, SPAMC

4. Kết quả thực nghiệm
C. Hiệu quả thực thi

- D500kC5T5N{500,600,700,800,900,1000)
- minSup = 0.1%
- 4 nodes

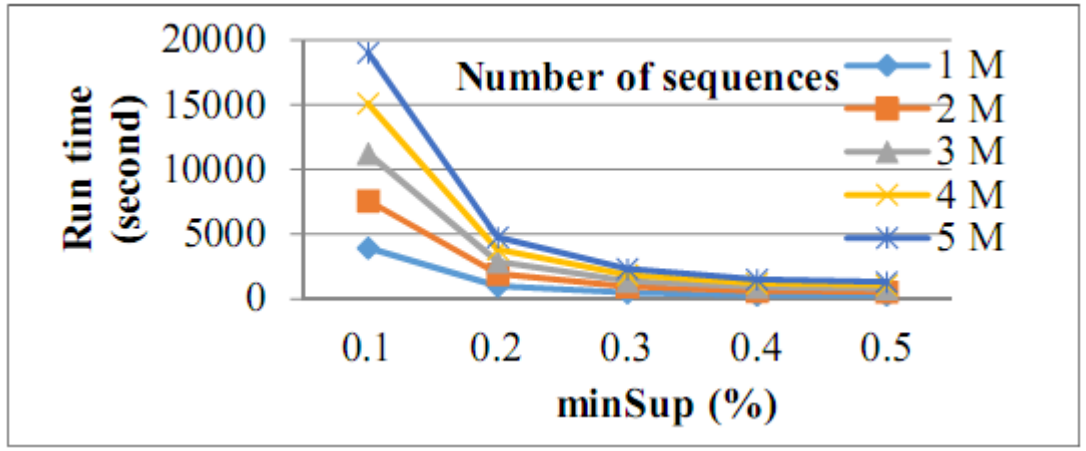


=> DSPDBV có thời gian thực thi tốt hơn SPAMC

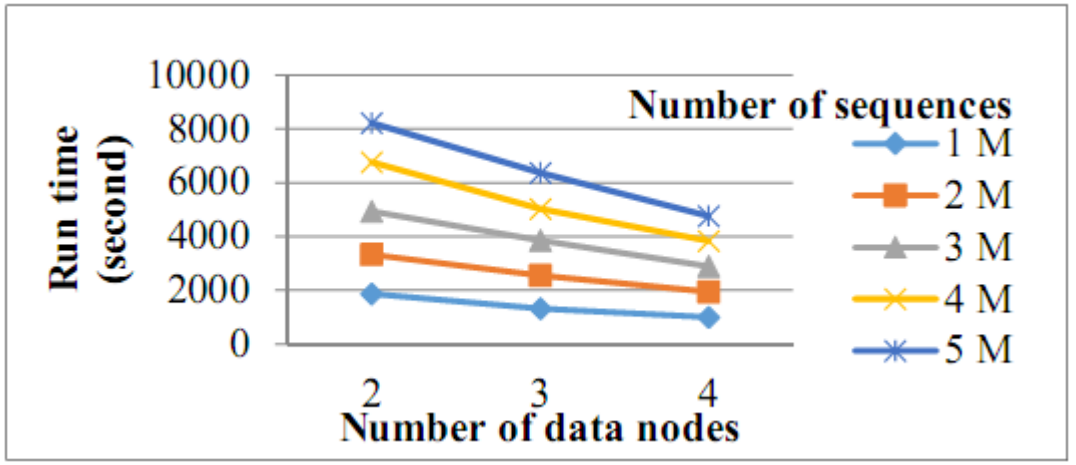
- Sử dụng cấu trúc bit động
- Loại bỏ sớm mẫu dư thừa
- Loại bỏ dữ liệu không cần thiết khi chuyển tiếp từ hàm Mapper sang hàm Reducer

4. Kết quả thực nghiệm
D. Tính mở rộng

Tập dữ liệu: D{1M,2M,3M,4M,5M}C5T5N1000



(a) Tác động của tập dữ liệu đối với hiệu quả khai thác



(b) Tác động của số lượng máy đối với hiệu quả khai thác

=> Thuật toán đề xuất đạt tính mở rộng cao trên tiêu chí ngưỡng hỗ trợ và số lượng máy trong Hadoop Cluster

5. Kết luận và Hướng phát triển
A. Kết luận

- Nghiên cứu tổng quan về bài toán khai thác chuỗi tuần tự từ CSDL chuỗi
- Trình bày ý tưởng và nội dung một phương pháp mới khai thác song song chuỗi từ CSDL chuỗi.
  - Khai thác hiệu quả trên CSDL chuỗi có kích thước và số lượng sự kiện phân biệt lớn
  - Đạt tính mở rộng cao
- Hạn chế
  - Chưa thực nghiệm trên các bộ dữ liệu thực lớn
  - Số lượng máy chạy thực nghiệm còn ít.

5. Kết luận và Hướng phát triển
B. Hướng phát triển

- Tiếp tục nghiên cứu cải tiến thuật toán DSPDBV
  - Lưu trữ
  - Loại bỏ sớm mẫu dư thừa
  - Áp dụng mô hình Streaming MapReduce
- Thực nghiệm
  - Dữ liệu thực có kích thước lớn
  - Số lượng máy chạy thực nghiệm nhiều hơn
- Phát triển thuật toán cho bài toán khai thác chuỗi tuần tự đóng.

Công trình công bố

1. *Distributed Algorithm for Sequential Pattern Mining on a Large Sequence Dataset*. **Tho. H, Bac. L, Thai. T.** 2017, KSE2017. In 9th International Conference on Knowledge and Systems Engineering. IEEE (accepted).

Tài liệu tham khảo

1. *MapReduce: Simplified Data Processing on Large Clusters.* **J. Dean and S. Ghemawat.** s.l. : Commun. ACM, 2008, Communications of the ACM 51 (1), pp. 107-113.

2. *"Apache Hadoop".* **The Apache Software Foundation.**

3. *Sequential PAttern Mining using A Bitmap Representation.* **Ayres. J, Gehrke. J, Yiu. T, Flannick. J.** 2002. in SIGKDD '02 Edmonton, Alberta, Canada.

4. *Fast Vertical Mining of Sequential Patterns Using Co-occurrence Infomation.* **Fournier-Viger, P., Gomariz, A., Campos, M., Thomas, R.** LNAI 8443, 2014, PAKDD 2014, pp. 40-52.

5. *DBV-Miner: a dynamic bit-vector approach for fast mining frequent closed itemsets.* **Vo, B., Hong, T.P., Le, B.** 2012, Expert Systems with Applications, 39, pp. 7196–7206.

6. *Combination of dynamic bit vectors and transaction information for mining frequent closed sequences efficiently.* **Tran, M., Le, B. and Vo, B.** 2015, Engineering Applications of Artificial Intelligence, 38, pp. 183-189.

7. *Highly scalable sequential pattern mining based on MapReduce model on the cloud.* **Chen CC, Tseng CY, Chen MS.** 2013, IEEE international congress on big data (BigData Congress'13), pp. 310–317

…