

Αλγόριθμοι Εξόρυξης Διαδικασιών στο Περιβάλλον Ανάπτυξης Spark

Βλάχης Πίτσιος – ΑΜ 220

Μεταπτυχιακό Πρόγραμμα Σπουδών

Ολοκληρωμένα Συστήματα Υλικού και Λογισμικού (ΟΣΥΛ)

Στόχος της διπλωματικής εργασίας

- Κατά πόσο είναι δυνατή η παραλληλοποίηση ενός αλγορίθμου εξόρυξης δεδομένων.
- Ποιο το όφελος από την παραλληλοποίηση του αλγόριθμου
- Αν αξίζει να παραλληλοποιείται ένας τέτοιος αλγόριθμος

Συνεισφορά της διπλωματικής εργασίας

- Μελέτη της θεωρίας της εξόρυξης διαδικασιών
- Μελέτη του αλγόριθμου Alpha και δικτύων Petri
- Μελέτη περιβάλλοντος ανάπτυξης εφαρμογών Apache Spark και γλώσσας προγραμματισμού Scala
- Υλοποίηση του αλγόριθμου Alpha σε περιβάλλον Apache Spark αλλά και σε απλή Scala
- Σύγκριση και αξιολόγηση πειραματικών μετρήσεων για διάφορα μεγέθη συστάδας υπολογιστών (cluster)

Εισαγωγή

- Τι είναι εξόρυξη διαδικασιών (**process mining**)?

Μέθοδος για την εξαγωγή μοντέλων από ένα σύνολο γεγονότων (event logs) που παράγονται από ένα σύστημα πληροφοριών.

Δεν υπάρχει αρχικό μοντέλο

- **Σκοπός**

Η κατανόηση αλλά και η βελτίωση της απόδοσης των διαδικασιών μιας επιχείρησης.

Ανακάλυψη μη-επιθυμητών διαδικασιών που ακολουθούνται στην πραγματικότητα

Παραγωγή μοντέλων συμπεριφοράς της λειτουργίας των επιχειρήσεων.

Παροχή **απαραίτητων εργαλείων** για την ανάλυση πραγματικών συμπεριφορών

Πρόταση λύσεων για την επίτευξη βέλτιστης λειτουργίας και αφαίρεση προβληματικών γεγονότων

Δομή των γεγονότων (event)

Συγκεκριμένη δομή με σίγουρη ύπαρξη των ιδιοτήτων

- caseId
- activity name
- Timestamp

Starting point for process mining:

Event data

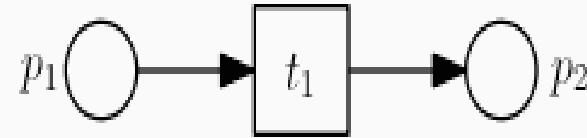
every row is an event
(here: an exam attempt)

student name	course name	exam date	mark
Peter Jones	Business Information systems	16-1-2014	8
Sandy Scott	Business Information systems	16-1-2014	5
Bridget White	Business Information systems	16-1-2014	9
John Anderson	Business Information systems	16-1-2014	8
Sandy Scott	BPM Systems	17-1-2014	7
Bridget White	BPM Systems	17-1-2014	8
Sandy Scott	Process Mining	20-1-2014	5
Bridget White	Process Mining	20-1-2014	9
John Anderson	Process Mining	20-1-2014	8
...

Μοντέλα Διαδικασιών

- Αποτέλεσμα ενός αλγορίθμου εξόρυξης διαδικασιών

Petri Net, BPMN, UML



- Δίκτυα Petri

Μεγάλη γκάμα συμπεριφορών όπως ακολουθίες (sequences), παραλληλισμό (concurrency), επιλογές (choices) και επαναλήψεις (loops).

Αποτελείται από κύκλους (κατάσταση p_1 και p_2) και τετράγωνα (μεταβάσεις t_1).

Ορίζεται σαν μια τριάδα (P, T, F) όπου P το σύνολο των καταστάσεων, T το σύνολο των μεταβάσεων και F το σύνολο των ακμών.

Αλγόριθμος Alpha

- Αλγόριθμος εύρεσης διαδικασιών

Διαβάζει σύνολο
γεγονότων

$$L_1 = [\langle a, b, c, d \rangle^3, \langle a, c, b, d \rangle^2, \langle a, e, d \rangle]$$

Παράγει ένα γράφημα (παράδειγμα δίκτυο Petri)

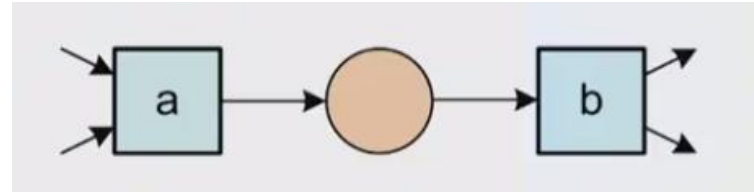
Δεν εξετάζει συχνότητες των δεδομένων, αλλά ακολουθίες από γεγονότα

Ίχνη δραστηριοτήτων (**traces** -> ακολουθία από events)

- Κατασκευάζει πίνακα σχέσεων (**footprint graph**)
 - ❑ **Direct succession** ($x > y$) Υπάρχουν ακολουθίες $\dots xy \dots$ (παράδειγμα $a > b$, $a > c$, $a > e$)
 - ❑ **Causality** ($x \rightarrow y$) Υπάρχουν ακολουθίες $\dots xy \dots$ Αλλά όχι $\dots yx \dots$ (παράδειγμα $a \rightarrow b$, $a \rightarrow c$, $a \rightarrow e$)
 - ❑ **Parallel** ($x \parallel y$) Υπάρχουν ακολουθίες $\dots xy \dots$ και $\dots yx \dots$ (παράδειγμα $b \parallel c$, $c \parallel b$)
 - ❑ **Choice- exclusiveness** ($x \# y$) Δεν υπάρχουν ακολουθίες $\dots xy \dots$ ούτε και $\dots yx \dots$ (παράδειγμα $a \# d$, $b \# e$)

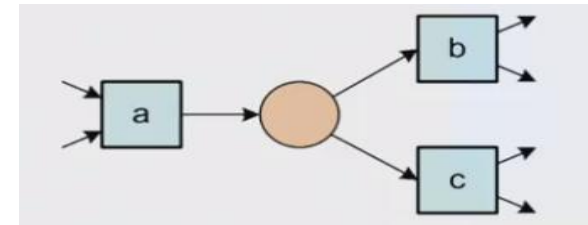
Αλγόριθμος Alpha – Δομικά Στοιχεία

- Causality pattern ($a \rightarrow b$)

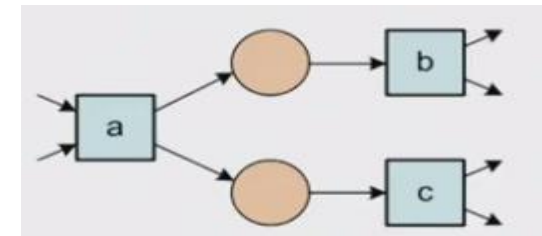


- XOR-split pattern ($a \rightarrow b$, $a \rightarrow c$, $b \# c$)

Επιλογή μεταξύ b και c



- AND-split pattern ($a \rightarrow b$, $a \rightarrow c$, $b || c$)



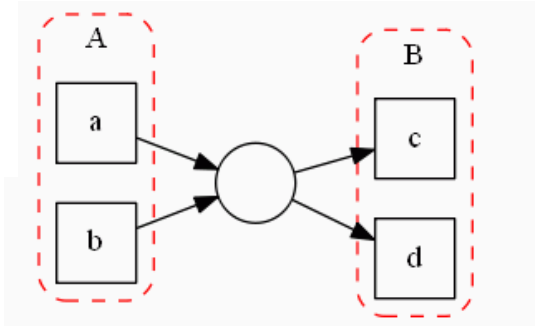
Αλγόριθμος Alpha – Βήματα (1)

1. Μοναδικές **μεταβάσεις (events)** που υπάρχουν στο σύνολο γεγονότων.
2. Όλες τις αρχικές **μεταβάσεις (events)** που υπάρχουν στο σύνολο γεγονότων, δηλαδή το πρώτο στοιχείο κάθε **trace** που ανήκει στο σύνολο γεγονότων.
3. Όλες τις τελικές **μεταβάσεις (events)** που υπάρχουν στο σύνολο γεγονότων, δηλαδή το τελευταίο στοιχείο κάθε **trace** που ανήκει στο σύνολο γεγονότων.
4. Κατασκευή πίνακα σχέσεων (**footprint graph**) και εύρεση **causal groups**

Causal group είναι group από σύνολα (A, B) για τα οποία ισχύει

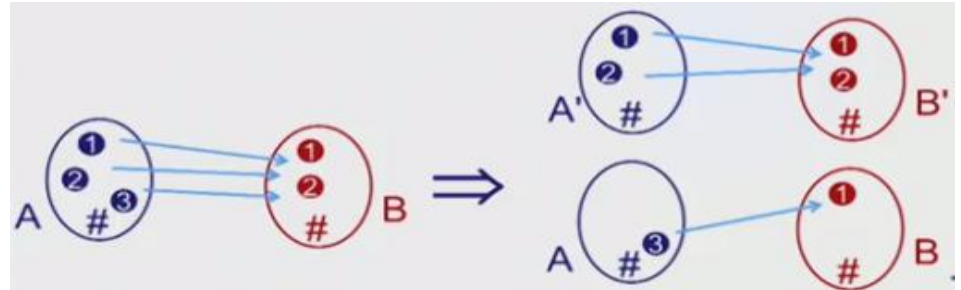
- Κάθε $t_A \in A$ πρέπει να συνδέεται με όλα τα $t_B \in B$ για κάθε **place** p
- $\forall a \in A$ και $\forall b \in B$ πρέπει να ισχύει $a \rightarrow b$
- $\forall a_1, a_2 \in A : a_1 \# a_2$ και $\forall b_1, b_2 \in B : b_1 \# b_2$

$a \# b$, $c \# d$, $a \rightarrow c$, $a \rightarrow d$, $b \rightarrow c$, $b \rightarrow d$



Αλγόριθμος Alpha – Βήματα (2)

5. Διαγραφή των μη-μέγιστων causal groups από το βήμα 4




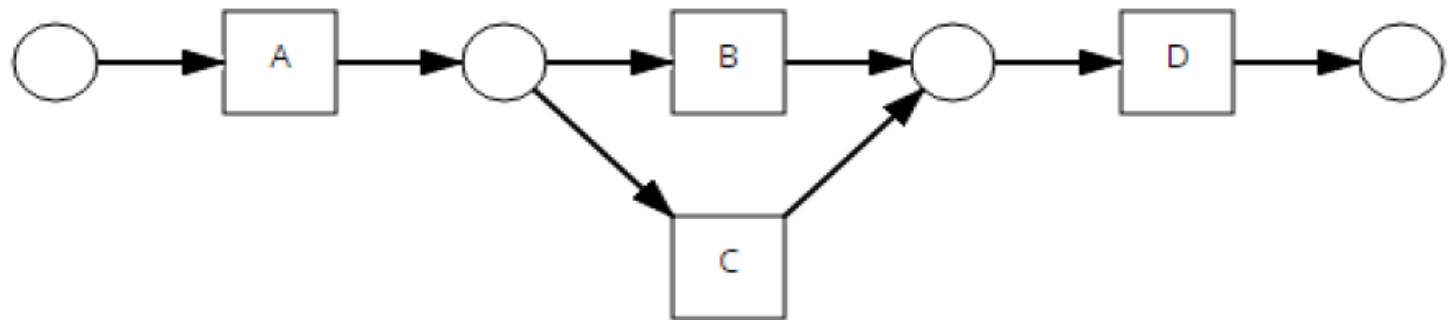
6. Εύρεση των καταστάσεων του δικτύου Petri από τα causal groups. Προσθήκη μιας αρχικής και μιας τελικής κατάστασης. Αποτελεί το σύνολο P της τριάδας (P, T, F) που περιγράφει το δίκτυο Petri.
7. Εύρεση των ακμών του δικτύου Petri (F από την τριάδα (P, T, F))
8. Κατασκευή του τελικού δικτύου Petri το οποίο αποτελείται από μια τριάδα συνόλων (P, T, F) .

Αλγόριθμος Alpha – Παράδειγμα

$W = \{ABD, ACD\}$

	A	B	C	D
A	#	→	→	#
B	←	#	#	→
C	←	#	#	→
D	#	←	←	#

- $T_w = \{A, B, C, D\}$
- $T_1 = \{A\}$
- $T_0 = \{D\}$
- $X_w = \{ \{A\} \times \{B\}, \{A\} \times \{C\}, \{A\}, \{B, C\}, \{B\} \times \{D\}, \{C\} \times \{D\}, \{B, C\}, \{D\} \}$
- $Y_w = \{ \{A\}, \{B, C\}, \{B, C\}, \{D\} \}$ 
- $P_w = \{P_{(\{A\}, \{B, C\})}, P_{(\{B, C\}, \{D\})}, i_w, o_w\}$
- $F_w = \{(i_w, A), (A, P_{(\{A\}, \{B, C\})}), (P_{(\{A\}, \{B, C\})}, B), (P_{(\{A\}, \{B, C\})}, C), (B, P_{(\{B, C\}, \{D\})}), (C, P_{(\{B, C\}, \{D\})}), (P_{(\{B, C\}, \{D\})}, D), (D, o_w)\}$
- $\alpha(W) = (P_w, T_w, F_w)$



Apache Spark Framework

- Κατανεμημένη επεξεργασία δεδομένων σε κόμβους μιας συστάδας υπολογιστών (cluster)
- 10 έως 100 φορές γρηρότερο από το Hadoop's MapReduce
- APIs παρόμοια με δομές δεδομένων της Scala, της Java ή της Python
- αποθήκευση μεγάλου όγκου δεδομένων στην κύρια μνήμη, πολύ μεγάλη αύξηση της απόδοσης
- **Δομές δεδομένων του Spark**
resilient distributed dataset (RDD) , Dataset, Dataframe
- **Κοινόχρηστες μεταβλητές (shared variables)**
broadcast μεταβλητές και συσσωρευτές (accumulators)

Υλοποίηση του Αλγόριθμου Alpha (1)

- 2 εκδόσεις του αλγόριθμου Alpha (**Spark** και **non-Spark**)
- Παράμετροι
 - val** logPath = "src/main/resources/data.csv"
 - val** numOfTraces = 3
 - val** percentage : Float = 1
 - val** readAll : Boolean = **false**
 - val** filtering : Boolean = **true**
- Σχηματισμός traces στην μορφή **Dataset[(String, List[String])]** μόνο όσων είναι σε status **Completed** ταξινομημένα σύμφωνα με το **starttime** και ομαδοποιημένα σύμφωνα με το **orderID**.
- Εύρεση των σχέσεων **direct succession** σύμφωνα με το tuple
 - (AB, PairNotation(DIRECT, FOLLOW)) => (A>B)
 - (AB, PairNotation(INVERSE, FOLLOW)) => (B>A)
- events (A,B,C,D,E) => (AA, AB, AC, AD, BB, BC, BD, CC, CD, DD)

Υλοποίηση του αλγόριθμου Alpha (2)

- Εύρεση των σχέσεων
- Υπολογισμός πίνακα σχέσεων

$a \rightarrow b \text{ iff } a > b \ \&\& \ ! (b > a)$ (Relation.CAUSALITY)

$a \parallel b \text{ iff } a > b \ \&\& \ b > a$ (Relation.PARALLEL)

$a \# b \text{ iff } \! (a > b) \ \&\& \ \! (b > a)$ (Relation.NEVER_FOLLOW)

Footprint graph

- Εύρεση των Causal groups

1. Εύρεση όλων των μοναδικών events από κάθε πλευρά όλων των causal σχέσεων. $\{a,b\}$ και $\{c,d\}$

$a \# b$

2. Εύρεση όλων των δυνατών συνδυασμών για κάθε ένα από τα παραπάνω σύνολα $\text{List}[\{a,b\} / \{a\} / \{b\}]$ και $\text{List}[\{c,d\} / \{c\} / \{d\}]$

$c \# d$

$a \rightarrow c$

$a \rightarrow d$

$b \rightarrow c$

$b \rightarrow d$

3. Διαγραφή των συνόλων αυτών τα οποία έχουν έστω μια μη-NeverFollow (choice relation) σχέση

4. Σύνδεση όλων των συνόλων από τις δύο λιστές μόνο αν όλα τα στοιχεία των δύο λιστών είναι σε causal σχέση μεταξύ τους και σχηματισμός των causal groups

Χρήση broadcast μεταβλητών

Υλοποίηση του αλγόριθμου Alpha (3)

- Υπολογισμός μόνο των μέγιστων causal groups

Χρήση των συσσωρευτών (accumulators) και των broadcast μεταβλητών του Spark

- Εύρεση των καταστάσεων. Μετατροπή από μέγιστα causal groups

Κλάση State η οποία διαθέτει δύο παραμέτρους (είσοδοι/έξοδοι)

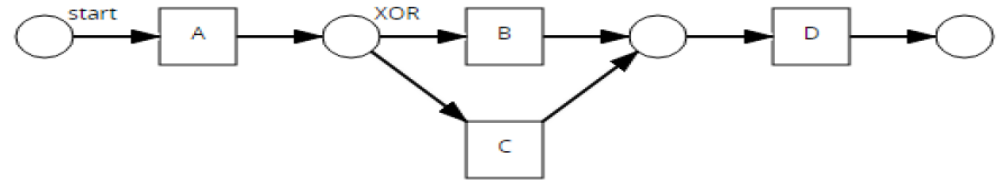
- Υπολογισμός των τόξων (arcs) με κλάση Edge που διαθέτει boolean flag

Direct == **true** -> (event, State)

Inverse == **false** -> (State, event)

Επιβεβαίωση Ορθής Λειτουργίας

- συνολικά 34 unit tests (end to end και για κάθε κλάση) Για end-2-end χρήση 5 παραδειγμάτων που υπάρχει γνώση του παραγόμενου δικτύου Petri



```
test( testName = "Check Alpha Algorithm functionality - Log 2") {  
  val logPath = "src/test/resources/log2.txt"  
  val traceTools: TraceTools = new TraceTools()  
  val tracesDS : Dataset[(String, List[String])] = traceTools.tracesDSFromLogFile(logPath)  
  val petriNet = AlphaAlgorithm.executeAlphaAlgorithm(tracesDS)  
  
  //check edges  
  assert(petriNet.getEdges().contains(new Edge( event = "A", new State(Set("A"), Set("B", "C")), direct = true)))  
  assert(petriNet.getEdges().contains(new Edge( event = "B", new State(Set("A"), Set("B", "C")), direct = false)))  
  assert(petriNet.getEdges().contains(new Edge( event = "C", new State(Set("A"), Set("B", "C")), direct = false)))  
  assert(petriNet.getEdges().contains(new Edge( event = "B", new State(Set("B", "C"), Set("D")), direct = true)))  
  assert(petriNet.getEdges().contains(new Edge( event = "C", new State(Set("B", "C"), Set("D")), direct = true)))  
  assert(petriNet.getEdges().contains(new Edge( event = "D", new State(Set("B", "C"), Set("D")), direct = false)))  
  
  //check initial edges  
  assert(petriNet.getEdges().contains(new Edge( event = "A", new State(Set.empty, Set("A")), direct = false)))  
  
  //check final edges  
  assert(petriNet.getEdges().contains(new Edge( event = "D", new State(Set("D"), Set.empty), direct = true)))  
  
  //check wrong directionality  
  assert(!petriNet.getEdges().contains(new Edge( event = "B", new State(Set("B", "C"), Set("D")), direct = false)))  
  assert(!petriNet.getEdges().contains(new Edge( event = "D", new State(Set("B", "C"), Set("D")), direct = true)))  
  
  assert(petriNet.getEdges().size==8)  
}
```


Εκτέλεση Αλγορίθμου – Μετρήσεις (1)

- **Databricks.** Προσφέρει διαχείριση συστάδας υπολογιστών (cluster) και εκτέλεση κώδικα με notebooks
- 2 notebooks (Spark και non-Spark version του αλγόριθμου Alpha)
- **Μετρήσεις**
 - Εκτέλεση non-Spark έκδοσης του αλγόριθμου Alpha στο Databricks Community Edition (1 κεντρικό κόμβο χωρίς κανένα βοηθητικό κόμβο).
 - Εκτέλεση της Spark έκδοσης του αλγόριθμου Alpha στο Databricks Community Edition (1 κεντρικό κόμβο χωρίς κανένα βοηθητικό κόμβο).
 - Εκτέλεση της Spark έκδοσης του αλγόριθμου Alpha στο Databricks με 1 κεντρικό κόμβο και 2 βοηθητικούς κόμβους.
 - Εκτέλεση της Spark έκδοσης του αλγόριθμου Alpha στο Databricks με 1 κεντρικό κόμβο και 4 βοηθητικούς κόμβους.

Σύνολο δεδομένων

Δεδομένα εισόδου: σύνολο δεδομένων σε csv αρχείο με μέγεθος 400mb με
36100 traces

Schema:

col_name	data_type
orderid	string
eventid	string
eventname	string
eventnameFIX	string
eventclass	string
type	string
system	string
descr	string
stage	string
actor	string
status	string
istatus	string
health	string
starttime	string
endtime	string
duration	string
reference_id	string

orderid	eventname	starttime	endtime	status
1-44547712794	BillingActivationFunction	2018-05-14 20:47:19	2018-05-14 20:48:20	Completed
1-44547712794	CompleteUpdateFunction	2018-05-14 20:48:25	2018-05-14 20:48:27	Completed
1-44547712794	FulfillBillingOrderUpdateFunction	2018-05-14 20:48:20	2018-05-14 20:48:25	Completed
1-44547712794	FulfillProvisionFunction	2018-05-14 19:50:45	2018-05-14 20:45:05	Completed
1-44547712794	ProvisionOrderUpdateFunction	2018-05-14 20:45:05	2018-05-14 20:45:05	Completed
1-44547712794	ProvisionUpdateFunction	2018-05-14 20:42:29	2018-05-14 20:42:30	Completed
1-44547712794	SyncCustomerOrderUpdateFunction	2018-05-14 19:50:45	2018-05-14 19:50:45	Completed
1-44547842055	BillingActivationFunction	2018-05-15 11:42:10	2018-05-15 11:43:10	Completed
1-44547842055	CompleteUpdateFunction	2018-05-15 11:43:21	2018-05-15 11:43:25	Completed
1-44547842055	FulfillBillingOrderUpdateFunction	2018-05-15 11:43:10	2018-05-15 11:43:21	Completed
1-44547842055	FulfillProvisionFunction	2018-05-14 19:58:59	2018-05-15 11:40:07	Completed
1-44547842055	ProvisionOrderUpdateFunction	2018-05-15 11:40:07	2018-05-15 11:40:07	Completed
1-44547842055	ProvisionUpdateFunction	2018-05-15 11:36:10	2018-05-15 11:36:12	Completed
1-44547842055	SyncCustomerOrderUpdateFunction	2018-05-14 19:58:59	2018-05-14 19:58:59	Completed
1-44547852652	BillingActivationFunction	2018-05-14 20:56:29	2018-05-14 20:57:31	Completed
1-44547852652	CompleteUpdateFunction	2018-05-14 20:57:34	2018-05-14 20:57:35	Completed
1-44547852652	FulfillBillingOrderUpdateFunction	2018-05-14 20:57:31	2018-05-14 20:57:34	Completed
1-44547852652	FulfillProvisionFunction	2018-05-14 20:03:19	2018-05-14 20:55:04	Completed

Εκτέλεση Αλγορίθμου – Μετρήσεις (1)

Πίνακας 1

percentage = 2

readAll = true

filtering = true

Αρχικός αριθμός από traces = 36094

Αριθμός traces προς επεξεργασία = 7

Συνολικός αριθμός από events = 16

	Non-Spark	Spark (1 κεντρικό – 0 βοηθητικούς κόμβους)	Spark (1 κεντρικό – 2 βοηθητικούς κόμβους)	Spark (1 κεντρικό – 4 βοηθητικούς κόμβους)
Χρόνος εκτέλεσης	2.60 λεπτά	3.60 λεπτά	1.54 λεπτά	1.2 λεπτά

Πίνακας 3

percentage = 0.8

readAll = true

filtering = true

Αρχικός αριθμός από traces = 36094

Αριθμός traces προς επεξεργασία = 9

Συνολικός αριθμός από events = 17

	Non-Spark	Spark (1 κεντρικό – 0 βοηθητικούς κόμβους)	Spark (1 κεντρικό – 2 βοηθητικούς κόμβους)	Spark (1 κεντρικό – 4 βοηθητικούς κόμβους)
Χρόνος εκτέλεσης	10.35 λεπτά	8.17 λεπτά	6.07 λεπτά	3.35 λεπτά

Εκτέλεση Αλγορίθμου – Μετρήσεις (2)

Πίνακας 4

percentage = 0.7

readAll = true

filtering = true

Αρχικός αριθμός από traces = 36094

Αριθμός traces προς επεξεργασία = 10

Συνολικός αριθμός από events = 18

	Non-Spark	Spark (1 κεντρικό – 0 βοηθητικούς κόμβους)	Spark (1 κεντρικό – 2 βοηθητικούς κόμβους)	Spark (1 κεντρικό – 4 βοηθητικούς κόμβους)
Χρόνος εκτέλεσης	34.60 λεπτά	17.35 λεπτά	15.40 λεπτά	9.92 λεπτά

Πίνακας 5

percentage = 0.5

readAll = true

filtering = true

Αρχικός αριθμός από traces = 36094

Αριθμός traces προς επεξεργασία = 12

Συνολικός αριθμός από events = 19

	Non-Spark	Spark (1 κεντρικό – 0 βοηθητικούς κόμβους)	Spark (1 κεντρικό – 2 βοηθητικούς κόμβους)	Spark (1 κεντρικό – 4 βοηθητικούς κόμβους)
Χρόνος εκτέλεσης	2.75 ώρες	1.06 ώρες	52.25 λεπτά	43.58 λεπτά

Συμπεράσματα

- **Spark** έκδοση του αλγόριθμου Alpha γρηγορότερη από την **non-Spark** έκδοση (μεγαλύτερη διαφορά 2.75 ώρες έναντι 44 λεπτών)
- **Εκτός** από τον πίνακα 1. **Non-Spark** έκδοση γρηγορότερη από **Spark** έκδοση για cluster με κανένα κόμβο.
- **Λόγοι**
 - το **Databricks Community Edition** παρέχει cluster μόνο με ένα κεντρικό κόμβο (χωρίς κανένα βοηθητικό κόμβο)
 - Τα δεδομένα αποθηκεύονται στην μνήμη RAM του συστήματος
 - Χρόνος για την προετοιμασία της εκτέλεσης ενός Spark προγράμματος
 - Serialization / deserialization δεδομένων

Προοπτικές

- Χρήση του Spark Streaming. Μελέτη συμπεριφοράς του αλγόριθμου Alpha όταν εμφανίζονται δεδομένα σε πραγματικό χρόνο
- Υλοποίηση άλλων αλγόριθμων εξόρυξης διαδικασιών όπως ο Flexible Heuristic Miner
- Υλοποίηση του αλγόριθμου Alpha σε περιβάλλον ανάπτυξης εφαρμογών Apache Flink

Ευχαριστίες

κ. Ζαρολιάγκης

κ. Σιούτας

κ. Τζήμας

κ. Βιέννας