US 20150262064A1

(19) United States
(12) Patent Application Publication
Burger et al.
(10) Pub. No.: US 2015/0262064 A1
(43) Pub. Date: Sep. 17, 2015

(54) PARALLEL DECISION TREE PROCESSOR ARCHITECTURE

(71) Applicant: **MICROSOFT CORPORATION**, REDMOND, WA (US)

(72) Inventors: **Douglas C. Burger**, Bellevue, WA (US); **James R. Larus**, Lusanne (CH); **Andrew Putnam**, Seattle, WA (US); **Jan Gray**, Bellevue, WA (US)

(73) Assignee: **MICROSOFT CORPORATION**, REDMOND, WA (US)

(21) Appl. No.: **14/216,990**

(22) Filed: **Mar. 17, 2014**

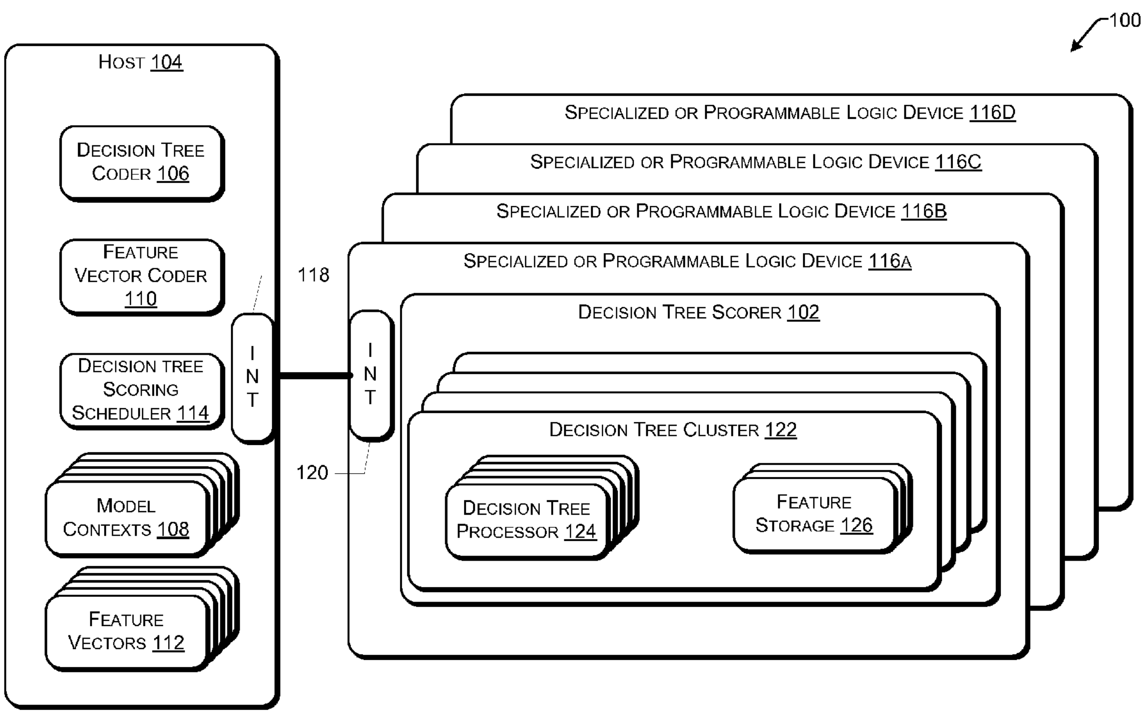
Publication Classification

(51) **Int. Cl.**
G06N 5/02 (2006.01)
G06F 17/30 (2006.01)

(52) **U.S. Cl.**
CPC *G06N 5/022* (2013.01); *G06F 17/30961* (2013.01)

(57) **ABSTRACT**

Disclosed herein is a decision tree multi-processor system. The system includes a plurality of decision tree processors that access a common feature vector and execute one or more decision trees with respect to the common feature vector.



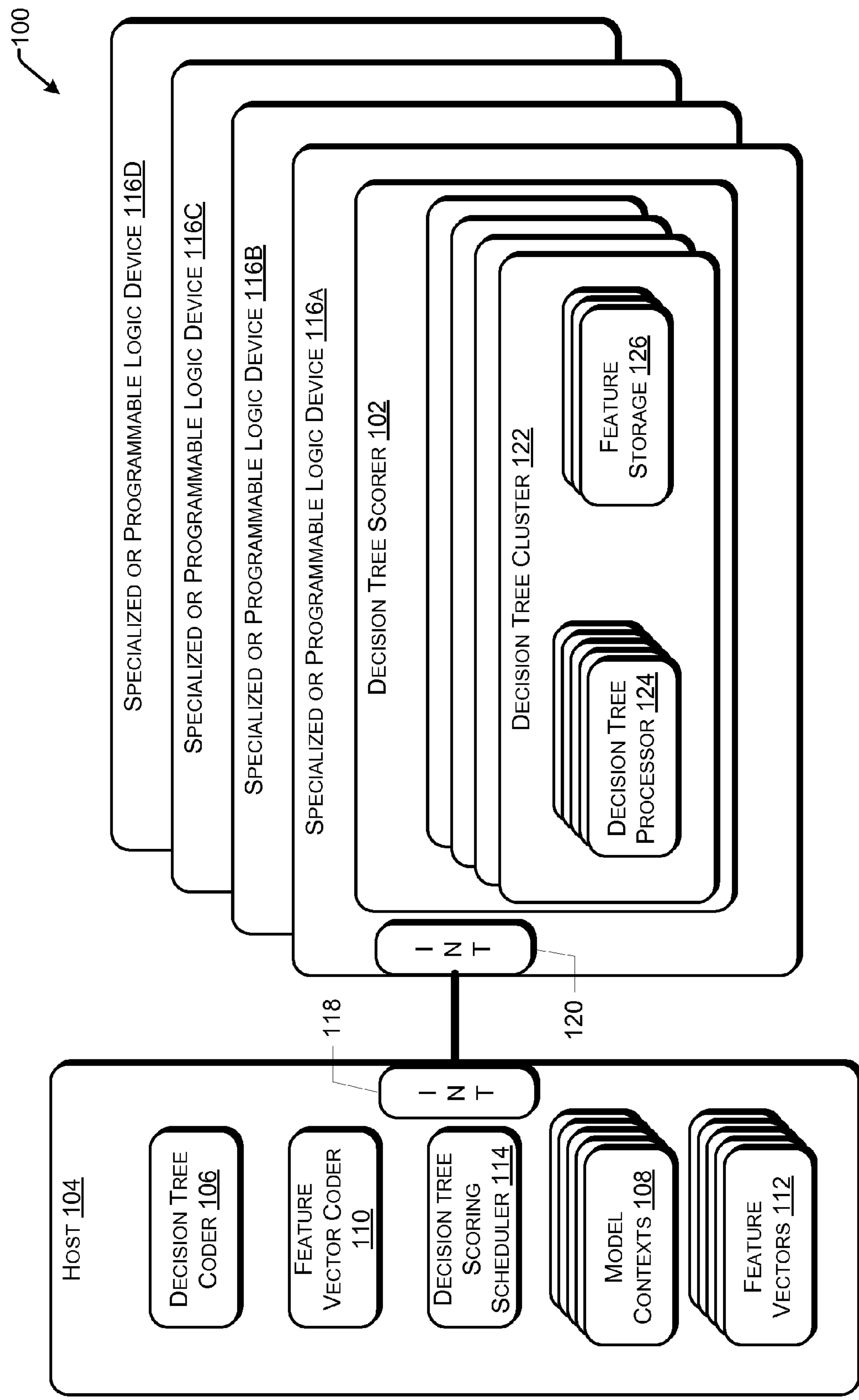

100
Specialized or Programmable Logic Device 116D
Specialized or Programmable Logic Device 116C
Specialized or Programmable Logic Device 116B
Specialized or Programmable Logic Device 116A
Decision Tree Scorer 102
Decision Tree Cluster 122
Feature Storage 126
Decision Tree Processor 124
INT
120
118
INT
Host 104
Decision Tree Coder 106
Feature Vector Coder 110
Decision Tree Scoring Scheduler 114
Model Contexts 108
Feature Vectors 112

FIG. 1

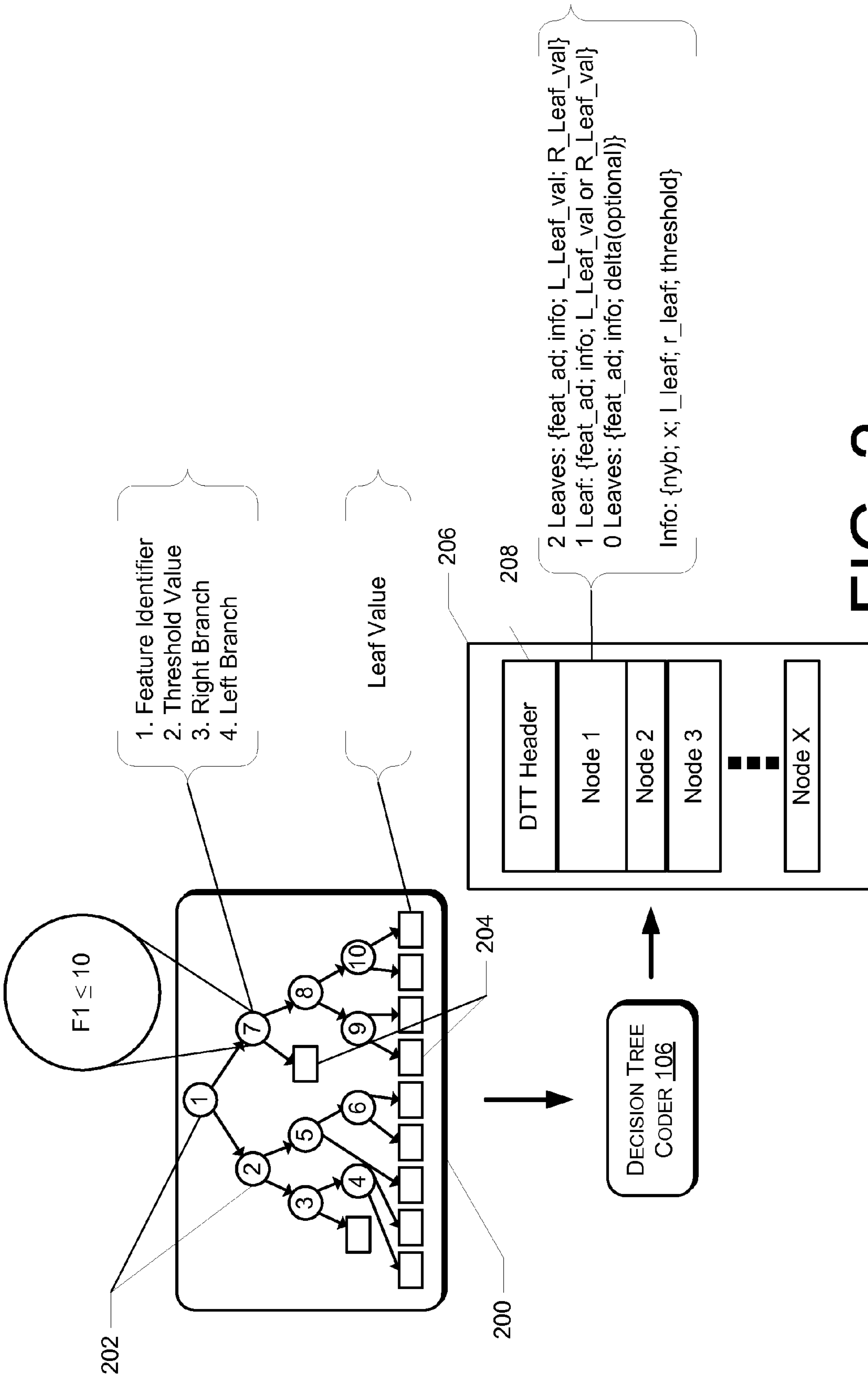

F1 ≤ 10
1. Feature Identifier
2. Threshold Value
3. Right Branch
4. Left Branch
Leaf Value
202
200
204
Decision Tree Coder 106
206
208
DTT Header
Node 1
Node 2
Node 3
Node X
2 Leaves: {feat_ad; info; L_Leaf_val; R_Leaf_val}
1 Leaf: {feat_ad; info; L_Leaf_val or R_Leaf_val}
0 Leaves: {feat_ad; info; delta(optional)}
Info: {nyb; x; l_leaf; r_leaf; threshold}

FIG. 2

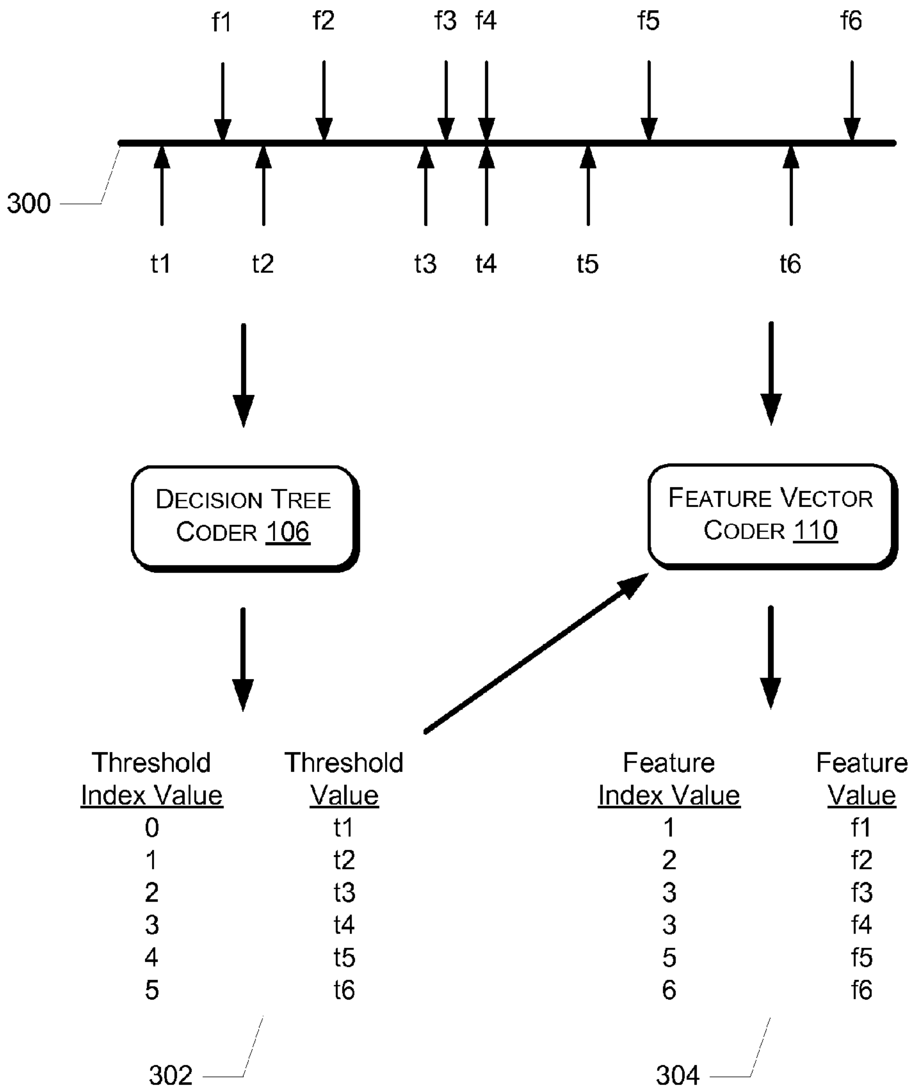

f1
f2
f3
f4
f5
f6
300
t1
t2
t3
t4
t5
t6
DECISION TREE CODER 106
FEATURE VECTOR CODER 110
Threshold Index Value
0
1
2
3
4
5
Threshold Value
t1
t2
t3
t4
t5
t6
Feature Index Value
1
2
3
3
5
6
Feature Value
f1
f2
f3
f4
f5
f6
302
304

FIG. 3

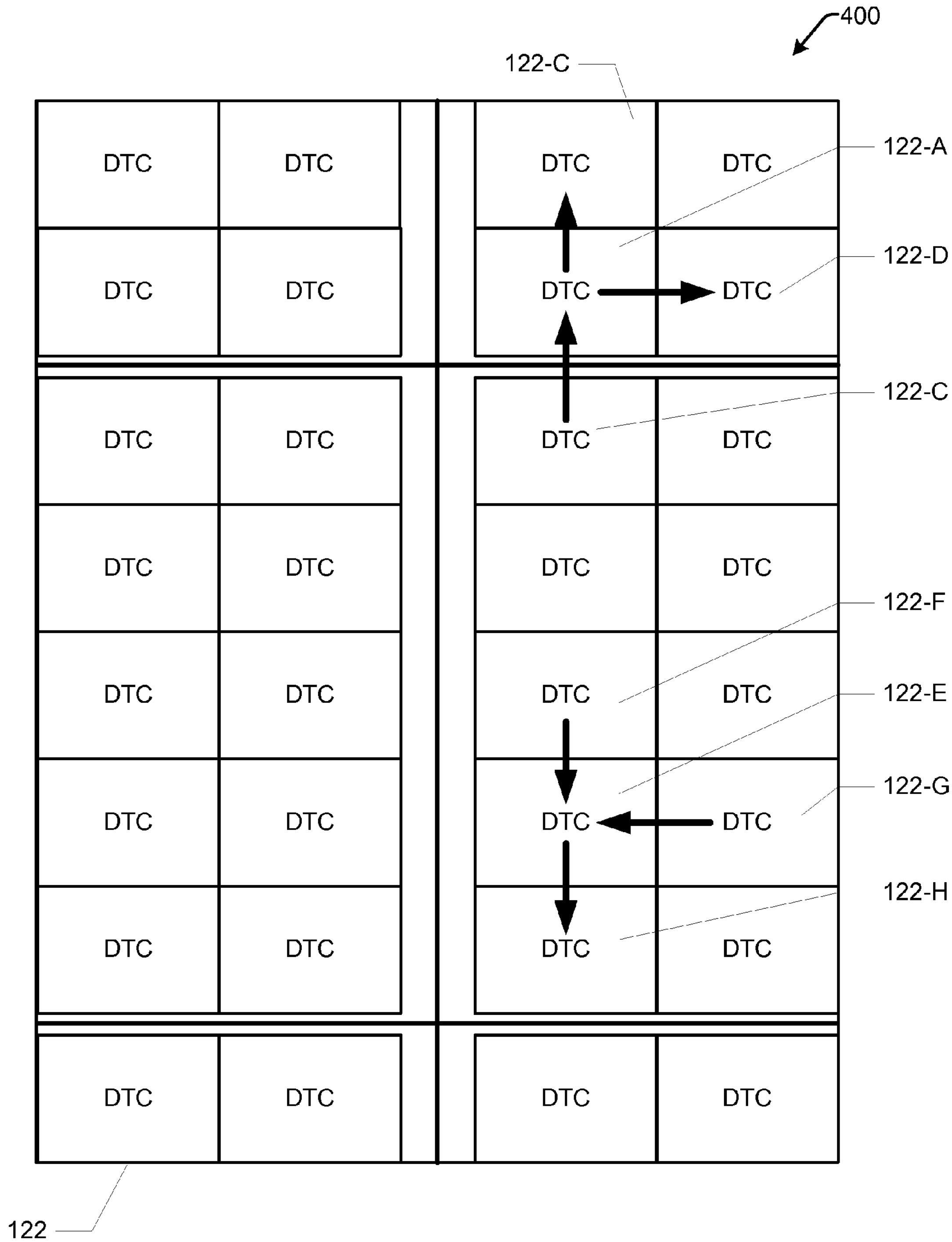

400
122-C
DTC
DTC
DTC
DTC
122-A
DTC
DTC
DTC
DTC
122-D
122-C
DTC
DTC
DTC
DTC
DTC
DTC
DTC
DTC
122-F
DTC
DTC
DTC
DTC
122-E
122-G
DTC
DTC
DTC
DTC
122-H
DTC
DTC
DTC
DTC
DTC
DTC
DTC
DTC
122

FIG. 4

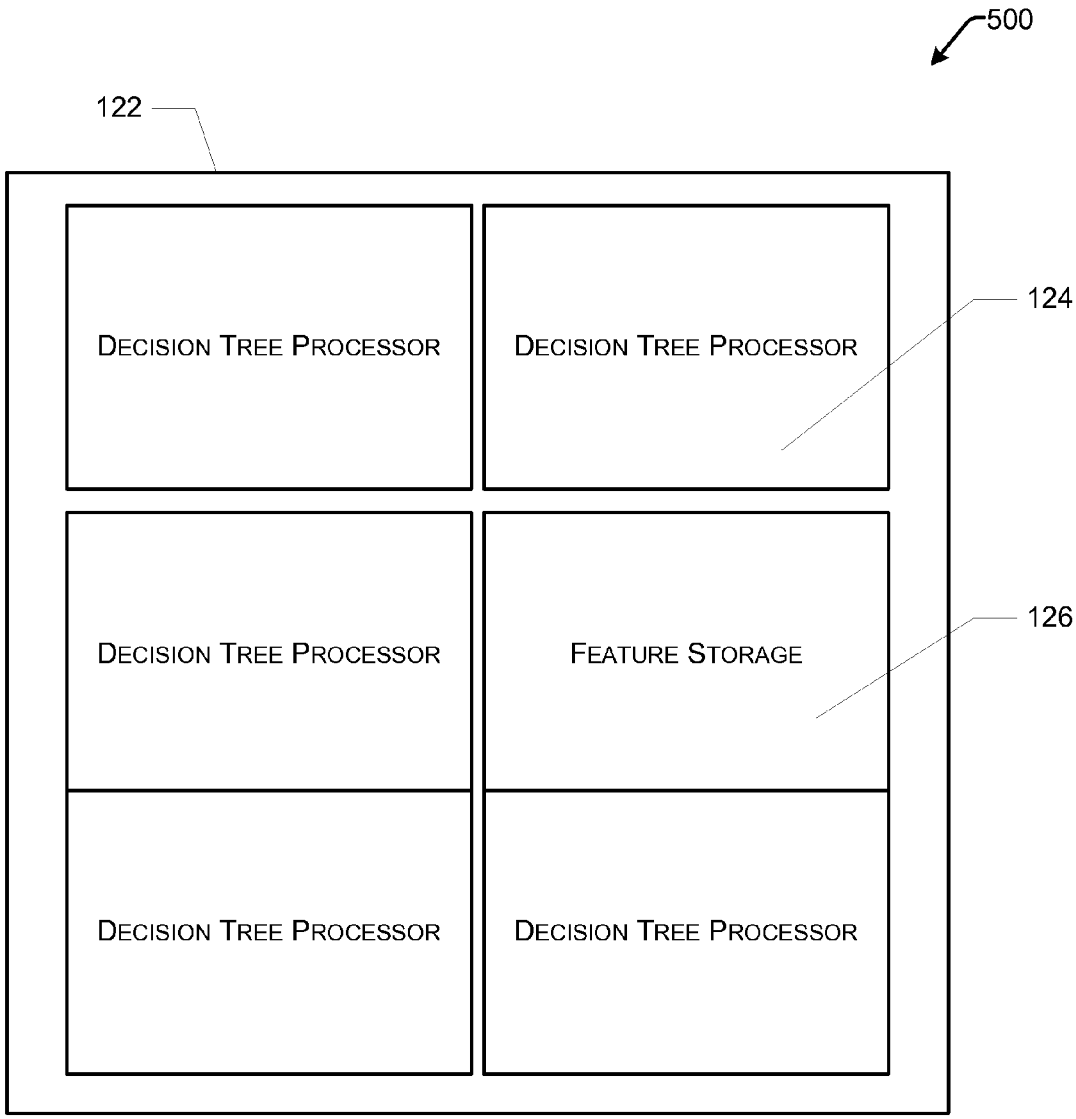
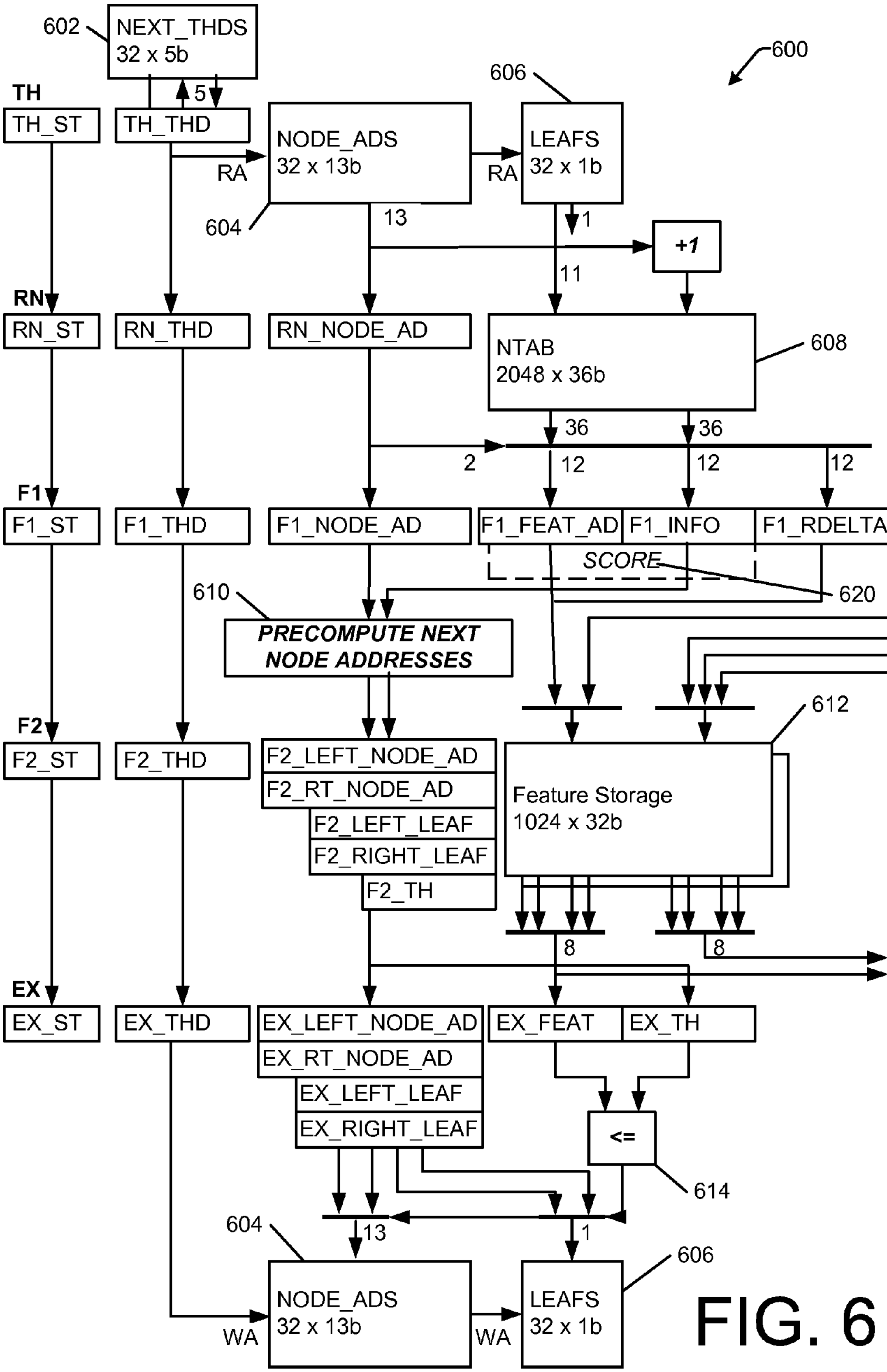

500
122
DECISION TREE PROCESSOR
DECISION TREE PROCESSOR
124
DECISION TREE PROCESSOR
FEATURE STORAGE
126
DECISION TREE PROCESSOR
DECISION TREE PROCESSOR

FIG. 5



600
602
NEXT_THDS
32 x 5b
5
TH
TH_ST
TH_THD
606
NODE_ADS
32 x 13b
RA
LEAFS
32 x 1b
RA
604
13
1
+1
11
RN
RN_ST
RN_THD
RN_NODE_AD
NTAB
2048 x 36b
608
36
36
2
12
12
12
F1
F1_ST
F1_THD
F1_NODE_AD
F1_FEAT_AD
F1_INFO
F1_RDELTA
SCORE
620
610
PRECOMPUTE NEXT NODE ADDRESSES
612
F2
F2_ST
F2_THD
F2_LEFT_NODE_AD
F2_RT_NODE_AD
F2_LEFT_LEAF
F2_RIGHT_LEAF
F2_TH
Feature Storage
1024 x 32b
8
8
EX
EX_ST
EX_THD
EX_LEFT_NODE_AD
EX_RT_NODE_AD
EX_LEFT_LEAF
EX_RIGHT_LEAF
EX_FEAT
EX_TH
<=
614
604
13
1
606
NODE_ADS
32 x 13b
WA
LEAFS
32 x 1b
WA

FIG. 6

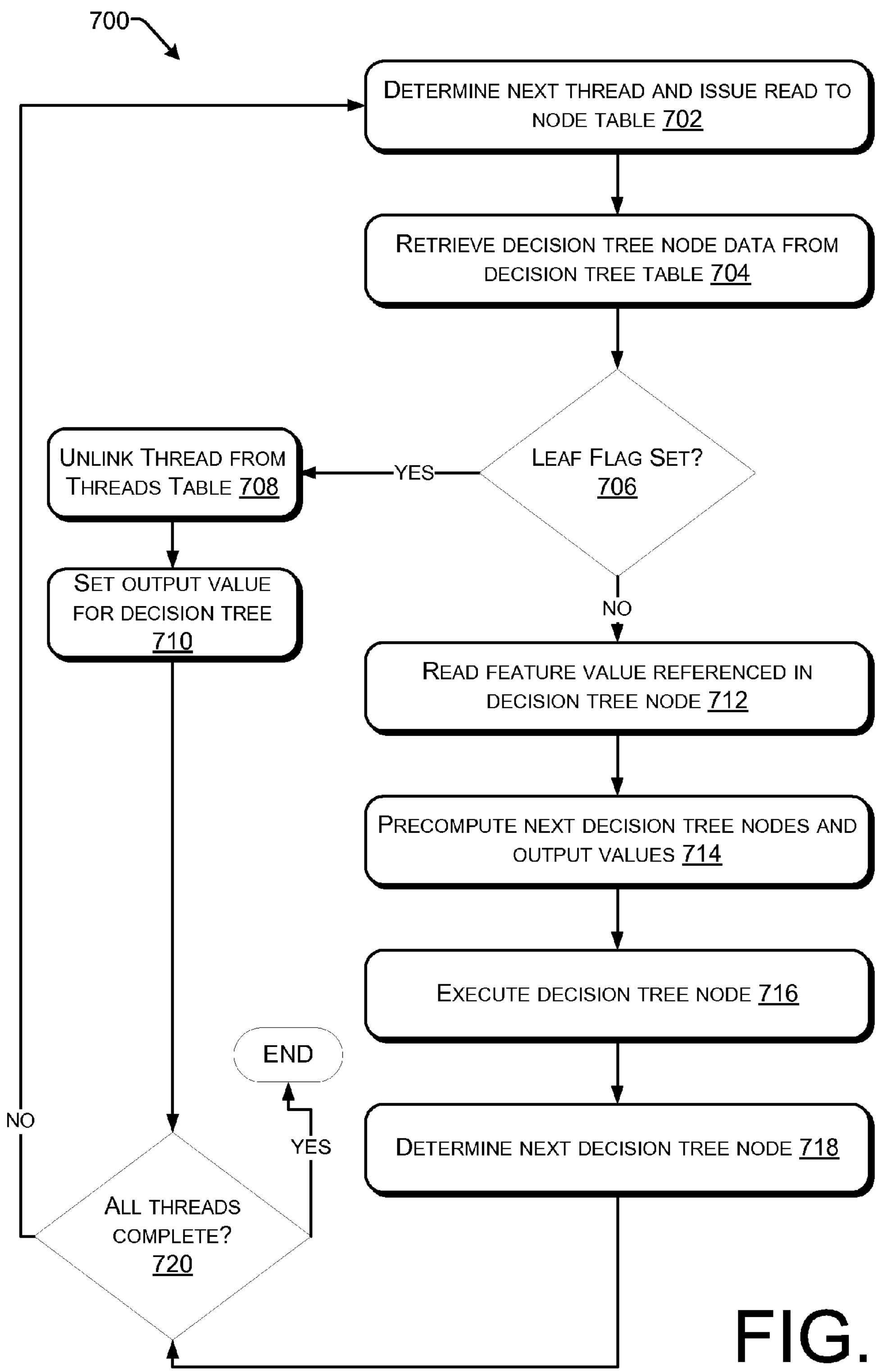

700
DETERMINE NEXT THREAD AND ISSUE READ TO NODE TABLE 702
RETRIEVE DECISION TREE NODE DATA FROM DECISION TREE TABLE 704
LEAF FLAG SET? 706
YES
UNLINK THREAD FROM THREADS TABLE 708
SET OUTPUT VALUE FOR DECISION TREE 710
NO
READ FEATURE VALUE REFERENCED IN DECISION TREE NODE 712
PRECOMPUTE NEXT DECISION TREE NODES AND OUTPUT VALUES 714
EXECUTE DECISION TREE NODE 716
END
DETERMINE NEXT DECISION TREE NODE 718
NO
YES
ALL THREADS COMPLETE? 720

FIG. 7

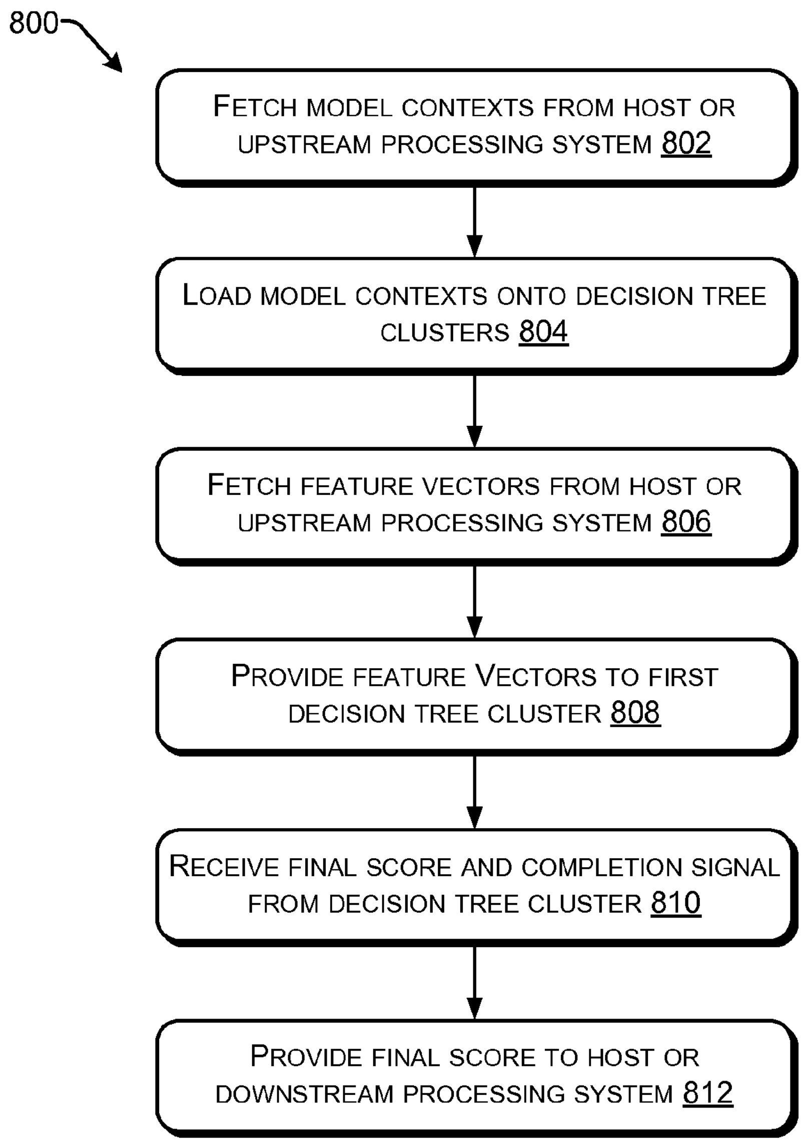

800
FETCH MODEL CONTEXTS FROM HOST OR UPSTREAM PROCESSING SYSTEM 802
LOAD MODEL CONTEXTS ONTO DECISION TREE CLUSTERS 804
FETCH FEATURE VECTORS FROM HOST OR UPSTREAM PROCESSING SYSTEM 806
PROVIDE FEATURE VECTORS TO FIRST DECISION TREE CLUSTER 808
RECEIVE FINAL SCORE AND COMPLETION SIGNAL FROM DECISION TREE CLUSTER 810
PROVIDE FINAL SCORE TO HOST OR DOWNSTREAM PROCESSING SYSTEM 812

FIG. 8

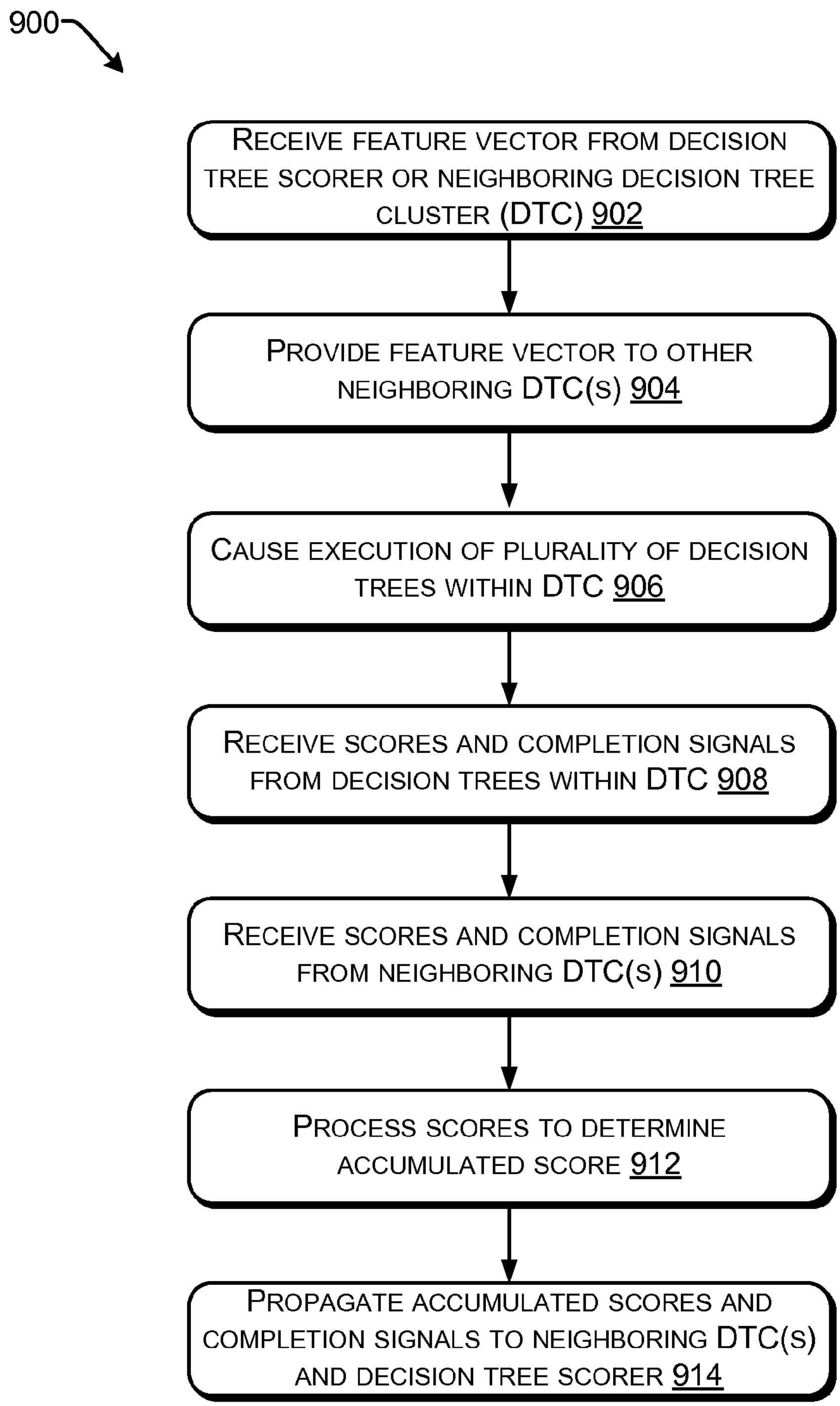

900
RECEIVE FEATURE VECTOR FROM DECISION TREE SCORER OR NEIGHBORING DECISION TREE CLUSTER (DTC) 902
PROVIDE FEATURE VECTOR TO OTHER NEIGHBORING DTC(S) 904
CAUSE EXECUTION OF PLURALITY OF DECISION TREES WITHIN DTC 906
RECEIVE SCORES AND COMPLETION SIGNALS FROM DECISION TREES WITHIN DTC 908
RECEIVE SCORES AND COMPLETION SIGNALS FROM NEIGHBORING DTC(S) 910
PROCESS SCORES TO DETERMINE ACCUMULATED SCORE 912
PROPAGATE ACCUMULATED SCORES AND COMPLETION SIGNALS TO NEIGHBORING DTC(S) AND DECISION TREE SCORER 914

FIG. 9

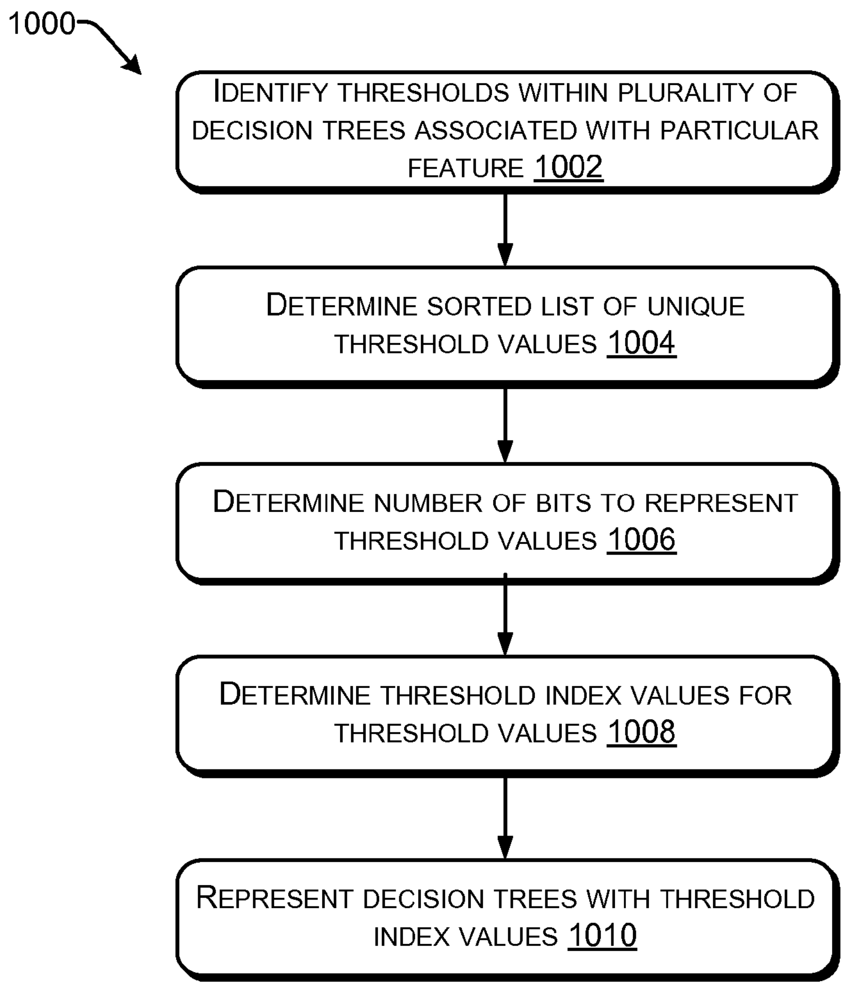

1000
IDENTIFY THRESHOLDS WITHIN PLURALITY OF DECISION TREES ASSOCIATED WITH PARTICULAR FEATURE 1002
DETERMINE SORTED LIST OF UNIQUE THRESHOLD VALUES 1004
DETERMINE NUMBER OF BITS TO REPRESENT THRESHOLD VALUES 1006
DETERMINE THRESHOLD INDEX VALUES FOR THRESHOLD VALUES 1008
REPRESENT DECISION TREES WITH THRESHOLD INDEX VALUES 1010

FIG. 10

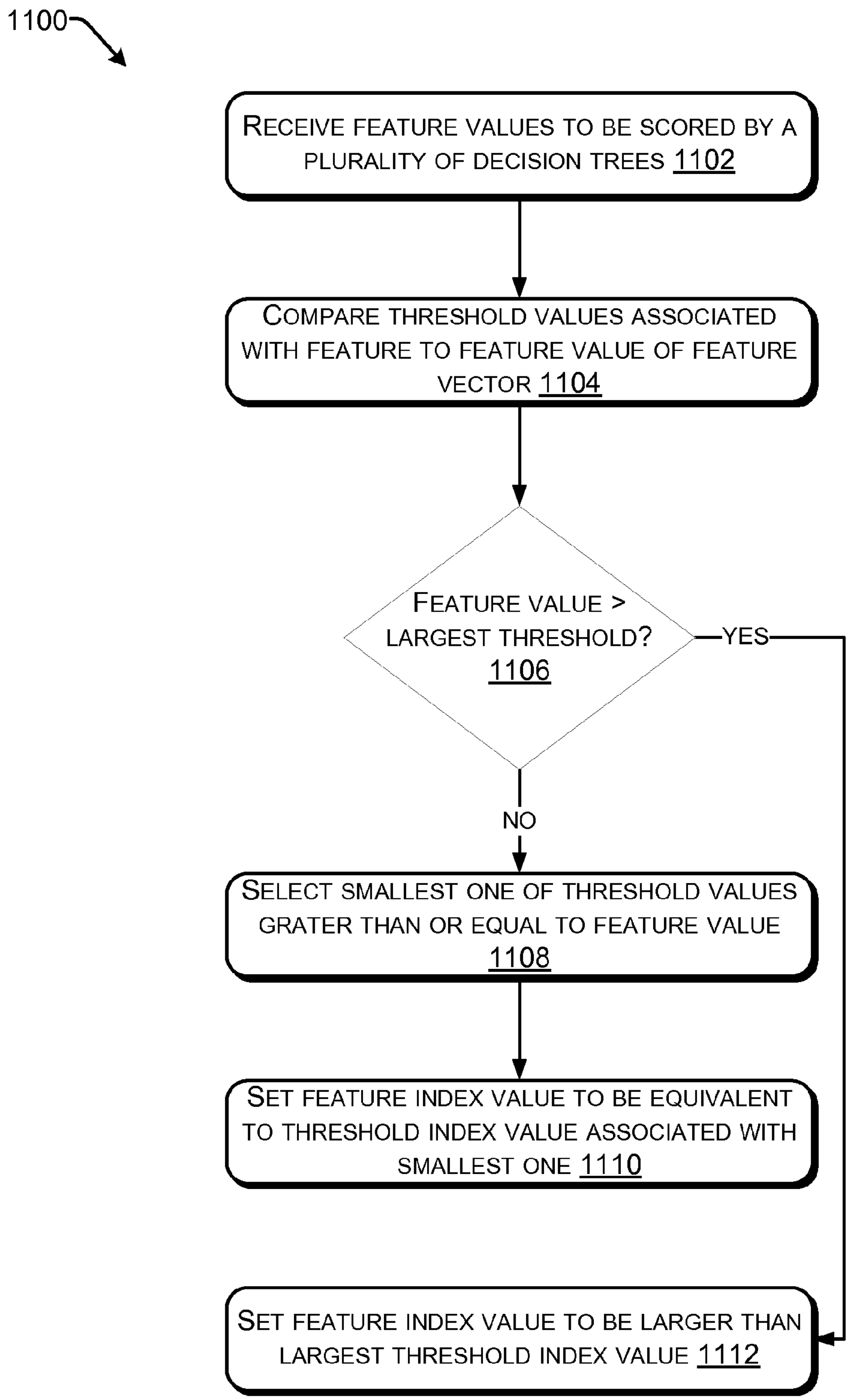

1100
RECEIVE FEATURE VALUES TO BE SCORED BY A PLURALITY OF DECISION TREES 1102
COMPARE THRESHOLD VALUES ASSOCIATED WITH FEATURE TO FEATURE VALUE OF FEATURE VECTOR 1104
FEATURE VALUE > LARGEST THRESHOLD? 1106
YES
NO
SELECT SMALLEST ONE OF THRESHOLD VALUES GRATER THAN OR EQUAL TO FEATURE VALUE 1108
SET FEATURE INDEX VALUE TO BE EQUIVALENT TO THRESHOLD INDEX VALUE ASSOCIATED WITH SMALLEST ONE 1110
SET FEATURE INDEX VALUE TO BE LARGER THAN LARGEST THRESHOLD INDEX VALUE 1112

FIG. 11

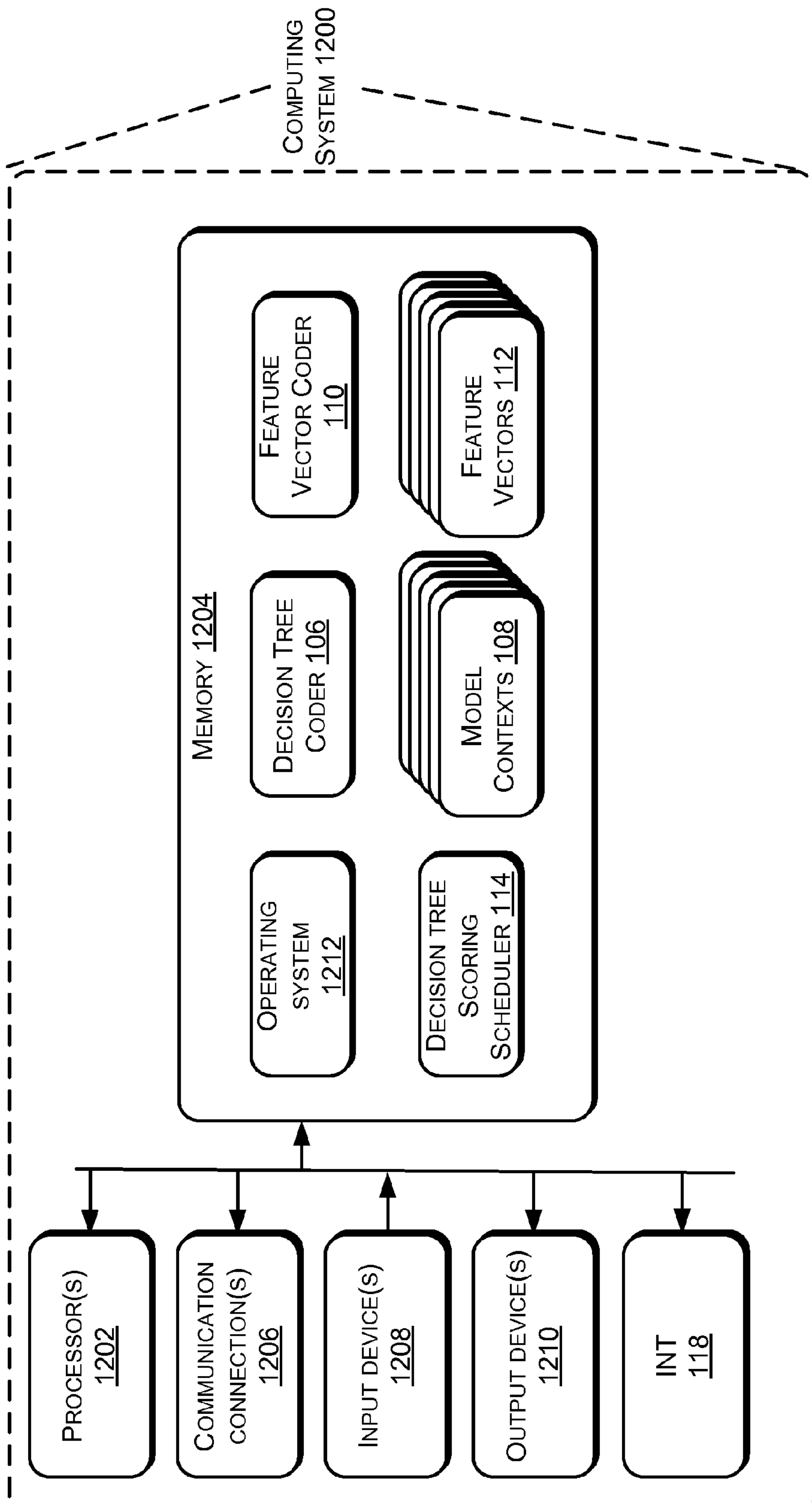

Computing System 1200
Memory 1204
Feature Vector Coder 110
Decision Tree Coder 106
Operating system 1212
Feature Vectors 112
Model Contexts 108
Decision Tree Scoring Scheduler 114
Processor(s) 1202
Communication connection(s) 1206
Input device(s) 1208
Output device(s) 1210
INT 118

FIG. 12

PARALLEL DECISION TREE PROCESSOR ARCHITECTURE

RELATED APPLICATIONS

[0001] The present application is related to concurrently filed U.S. application Ser. No. ______, entitled "Decision Tree Processors," the entire contents of which are hereby incorporated herein in their entirety. The present application is also related to concurrently filed U.S. application Ser. No. ______, entitled "Decision Tree Threshold Coding," the entire contents of which are hereby incorporated herein in their entirety.

BACKGROUND

[0002] A decision tree is a binary search tree comprised of decision nodes and left and right sub-trees and/or leaves. A decision node includes a decision to be made. Branches lead from a decision node to other decision nodes or to leaf nodes, and a selection of one of the branches is based on the decision made at the decision node. An example decision includes the comparison of two values, such as a feature value and a threshold value. If the feature value is less than or equal to the threshold value, then a left subtree is selected; if the feature value is not less than or equal to the threshold value, then the right subtree is selected. The branch is followed to the next node and, if the next node is a decision node, another decision is made, and so on until a branch leading to a leaf node is selected. A leaf node represents an output or an end-point of the decision tree. An example output is an output value, or a score, for the decision tree. This process is referred to as walking the decision tree.

[0003] Among other applications, decision trees are used to rank documents in document search. In one example, a decision tree is used to calculate the relevance of a particular item (e.g., a web page) to a particular search query. An initial set of candidate search result documents are obtained, and a feature vector for the candidate search result documents are produced. The feature vector represents various aspects (e.g., document statistics) of the candidate search result documents. One example of a feature is the number of times a search query word appears in the candidate document. Each decision tree node includes a threshold and a feature identifier, which can be used to look up the feature value for the candidate search result document. The decision tree is walked, and the tree-walking process eventually arrives at a leaf node and outputs the associated score. The score (or multiple scores if more than one decision tree is used) is used to determine the relevance of a candidate search result. The relative scores of multiple documents are used to rank the documents.

[0004] Besides search, decision trees have a variety of uses. Decision trees are used to implement gesture recognition, voice recognition, data mining, and other types of computations.

BRIEF SUMMARY

[0005] This Summary is provided in order to introduce simplified concepts of the present disclosure, which are further described below in the Detailed Description. This summary is not intended to identify essential features of the claimed subject matter, nor is it intended for use in determining the scope of the claimed subject matter.

[0006] Embodiments of the present description include hardware implementations of decision tree scoring, which enables faster decision tree scoring than conventional software-based decision tree scoring. On-chip architecture of the decision tree scoring system includes a plurality of decision tree processors implemented in parallel on one or more specialized or programmable logic circuits. At the top level of the on-chip architecture is a decision tree scorer (DTS) that receive feature vectors (e.g., sets of feature values) from an upstream computing system host or processing system, sends the feature vectors to a first decision tree cluster (DTC), receives scores from the decision tree clusters, and outputs the result to the host or other downstream system. At the next level of the hierarchy, a plurality of decision tree clusters (DTC) distributes feature vectors amongst themselves, and processes and propagates scores from decision tree processors to neighboring DTCs and to the DTS. The DTCs include one or more decision tree processors, and one or more feature storage tiles (FST). Feature value and threshold value compression reduce the bandwidth and storage requirements for the decision tree scoring system.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The Detailed Description is set forth with reference to the accompanying figures. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The use of the same reference numbers in different figures indicates similar or identical items.

[0008] FIG. **1** is a block diagram of an example decision tree scoring system that includes one or more hardware-implemented decision tree scorers in accordance with various embodiments.

[0009] FIG. **2** illustrates decision tree coding in accordance with various embodiments of the present disclosure.

[0010] FIG. **3** illustrates an example list of unique threshold values on a real number line.

[0011] FIG. **4** illustrates an example architecture of the decision tree scorer in accordance with various embodiments.

[0012] FIG. **5** illustrates an example architecture of a decision tree cluster in accordance with various embodiments.

[0013] FIG. **6** illustrates a multi-stage, multi-threaded, pipelined tree walking implementation of a decision tree processor, in accordance with various embodiments.

[0014] FIG. **7** depicts a flow graph that shows an example process of executing a decision tree node, in accordance with various embodiments.

[0015] FIG. **8** illustrates a process of scoring feature vectors a plurality of decision trees by a decision tree scorer, in accordance with various embodiments.

[0016] FIG. **9** illustrates a process of scoring a plurality of decision trees by decision tree clusters, in accordance with various embodiments.

[0017] FIG. **10** illustrates a process of coding threshold values of a plurality of decision trees in accordance with various embodiments.

[0018] FIG. **11** illustrates a process of coding a set of feature values, in accordance with various embodiments.

[0019] FIG. **12** is a block diagram of an example computing system usable to perform various methods described herein.

DETAILED DESCRIPTION

Overview

[0020] Embodiments of the present description include hardware implementations of decision tree scoring, which enables faster decision tree scoring than conventional software-based decision tree scoring. The hardware implementation includes one or more decision tree processors, implemented as circuitry, that execute decision tree programs. A decision tree program is a decision tree that has been converted to a program or other data executable by a decision tree processor. A decision tree program includes a decision tree table, which includes the various decision nodes, feature identifiers, threshold values, and output values for a decision tree. Some embodiments of decision tree processors walk decision trees in a multi-stage and/or multi-threaded fashion. In multi-threaded embodiments, each stage of the decision tree processor executes a different decision tree thread; thus an n-stage multi-threaded decision tree processor concurrently executes portions of up to n decision trees per cycle.

[0021] Embodiments include processes, systems, and apparatuses for coding, compressing, and/or compiling decision trees to be executed within a decision tree processor. In various embodiments, pointers are eliminated from decision trees by arranging the nodes such that some of the nodes with branches between them in the decision tree are adjacent to the particular node in the decision tree table. Other nodes are identified with next node data, such as offset or delta values. Leaf values are part of the decision tree node representations, rather than part of separate leaf node entries.

[0022] In some embodiments, feature value and threshold value compression reduces the bandwidth and storage requirements for a decision tree scoring system, while also increasing the size of workloads that embodiments are able to handle. In some embodiments of the present description, a sorted list is created for each threshold value that a particular feature is compared to in one or more decision trees, and threshold value indices are assigned to the threshold values. Although the total number of possible thresholds is high (e.g., represented in some embodiments by a 32-bit floating point number), the total number of actual thresholds for a particular feature in a plurality of decision trees is in practice much smaller, usually no more than 255 thresholds (although larger numbers of thresholds are possible). A dense or non-dense fixed-point small integer threshold value index is created. The threshold value index may be numbers from 0 to the total number of thresholds, and thus may be represented by a 4 bit, 8 bit, or other n-bit fixed point value. In other embodiments, the threshold value index may be negative numbers, and may include non-contiguous integer values, such as 0, 2, 4, 6, or other non-contiguous integer values. Feature values are also coded as n-bit fixed point feature value indices, such that comparisons of the threshold value indices to the feature value indices are equivalent to comparisons of the original, non-compressed threshold values to the original, non-compressed feature values.

[0023] In some embodiments, a plurality of decision tree processors is implemented in parallel on one or more specialized or programmable logic circuits. In some embodiments, the plurality of decision tree processors executes, or concurrently executes, decision trees with respect to a common feature vector. At the top level of the on-chip architecture is a decision tree scorer (DTS) that receives feature vectors (e.g., sets of feature values) from an upstream computing system host or processing system, sends the feature vectors to a first decision tree cluster (DTC), receives scores from the decision tree clusters, and outputs the result to the host or other downstream system. At the next level of the hierarchy, a plurality of decision tree clusters (DTC) distributes feature vectors amongst themselves and propagates scores from decision tree processors to neighboring DTCs and to the DTS. At the next level of the hierarchy, the DTCs include one or more decision tree processors, and one or more feature storage tiles (FST). The decision tree processors may be multi-threaded to concurrently execute multiple decision trees with respect to common feature vectors. The FST stores feature vectors to be scored against the plurality of decision trees, and in some embodiments are double-buffered to enable one set of features to be written to the FST while another set of features are accessed by the decision tree processors for scoring.

[0024] Embodiments described herein are amenable to implementation in specialized hardware such as in an ASIC, or in programmable logic device such as an FPGA. Various aspects of embodiments are also amenable to implementation in a multi-core processor, a system-on-chip (SoC) (e.g., one or more decision tree scoring cores on an SoC), and/or as a general purpose processor with an extended instruction set, and thus able to partially or wholly execute decision trees responsive to one or more atomic processor instructions. The devices, processes, and systems described herein may be implemented in a number of ways. Example implementations are provided below with reference to the following figures.

Example Decision Tree Scoring System

[0025] FIG. **1** is a block diagram of an example decision tree scoring system **100** that includes one or more hardware-implemented decision tree scorers **102** in accordance with various embodiments. A host **104** includes a decision tree coder **106** to code decision trees into model contexts **108** for execution on the decision tree scorers **102**. As described in more detail below, the decision tree coder **106** represents decision trees using variable-length nodes, wherein subtree pointers are eliminated with adjacencies and offsets, leaf values are included in the node representations, and threshold values are coded as threshold index values. The decision tree coder **106** reduces the sizes of the decision trees, to enable more of them to be loaded onto the decision tree scorer **102**. The decision tree coder **106** may also or alternatively compress the decision tree data (or coded decision tree data) of the model contexts **108** using other compression techniques. In these embodiments the decision tree scorer **102** or other on-chip logic is configured to decompress the compressed decision tree or coded decision tree data for scoring on the decision tree scorer **102**.

[0026] The host **104** also includes a feature vector coder **110** that codes feature values within feature vectors **112** to reduce the bandwidth and storage requirements of the decision tree scorers **102**, to make the feature vectors **112** compatible with the coded model contexts **108**, and to place the model contexts **108** and the feature vectors **112** into a form more easily processed by specialized hardware as described in various embodiments herein. As described in more detail elsewhere within this Detailed Description, the feature vector coder **110** selects feature index values for the features such that comparisons of the feature index values to threshold index values within the model contexts **108** are equivalent to comparisons of the corresponding feature values and threshold values.

[0027] The host **104** includes a decision tree scoring scheduler **114** that schedules decision tree scoring jobs. The host **104** receives or determines that various ones of the feature vectors **112** are to be scored against various ones of the model contexts **108**. An example set of decision tree scoring jobs includes:

[0028] Feature vector 1/Model Context A

[0029] Feature vector 2/Model Context B

[0030] Feature vector 3/Model Context A

[0031] Feature vector 4/Model Context B

[0032] Because it generally takes more time to load a new model context into the decision tree scorer **102** than it takes to load a new feature vector into the decision tree scorer **102**, the decision tree scoring scheduler **114** rearranges the decision scoring jobs to reduce the number of times that a new model context **108** is loaded into the decision tree scorer **102**. Continuing with the example above, the decision tree scoring jobs are rearranged as follows:

[0033] Feature vector 1/Model Context A

[0034] Feature vector 3/Model Context A

[0035] Feature vector 2/Model Context B

[0036] Feature vector 4/Model Context B

[0037] In the field of search, a model context is a set of decision trees associated with a type of search being performed. Examples of search contexts that utilize different sets of decision trees are language (search on English-language queries may be performed using a different model context that searches performed in German-language queries), image search, news search, video search, and so forth. Other search contexts may call for separate model contexts.

[0038] The host **104** is configured to be communicatively coupled to one or more specialized or programmable logic devices **116** via datapath interfaces, such as interfaces **118** and **120**. The interfaces **118** and **120** are, in various embodiments, Peripheral Component Interfaces Express (PCI-Express) interfaces, although other interface types and specifications may be used without departing from the scope of embodiments. The determination of the interface type may be based on interface bandwidth targets, which may in turn be based on the throughput targets for the decision tree scoring system **100**. In a particular example, where a target processing speed is one microsecond per search document scoring, using decision tree and feature compression techniques described herein results in a bandwidth target of approximately 2-8 KB per feature vector (e.g., per candidate search result document), or approximately 2-8 GB per second. PCI-Express is suitable for this target, although other interface types and specifications may also be suitable for this or other targets. Multiple interfaces may also be used in place of a single high-speed interface without departing from the scope of embodiments.

[0039] As described in more detail below, the host **104** may be implemented as a plurality of programming instructions executable by one or more general-purpose processors of a computing system. However, one or more aspects of the host **104** may be implemented on specialized or programmable logic circuits (such as ASIC chips or FPGA chips).

[0040] The decision tree scorer **102** includes one or more decision tree clusters **122**. The decision tree clusters **122** are configured to distribute the model contexts **108** and the feature vectors **112** amongst themselves. Alternatively, or in addition, the decision tree scorer **102** may include an interconnect network to pass the model contexts **108** and/or the feature vectors **112** throughout the decision tree scorer **102**. The decision tree clusters **122** are also configured to process and propagate decision tree scores from neighboring decision tree clusters **122**, as well as from the decision tree processors **124** within the decision tree clusters **122**. The decision tree clusters **122** are configured to process the scores received from the decision tree processors **124** and neighboring decision tree clusters—which may include summing the decision tree scores—and to propagate the processed scores (e.g., the summed scores) to other neighboring decision tree clusters **122**, as will be described in more detail elsewhere within this Detailed Description. The decision tree scorer **102** is configured to receive from one of the decision tree clusters **122** a final score (e.g., a scalar or a vector quantity) for the decision tree scoring job and to output the score to the host **104**, or another downstream device.

[0041] The decision tree processors **124** include circuitry to execute decision trees of one or more model contexts **108**, such as in parallel and concurrently against a common one of the feature vectors **112**, or against the different ones of the feature vectors **112**, depending on the implementation. Different ones of the feature storage **126** may store either a common one of the feature vectors **112** or different ones of the feature vectors **112**. The feature storage **126** within each decision tree cluster **122** may store the same or different ones of the feature vectors **112**.

[0042] As used herein, a decision tree processor **124** includes circuitry to score a decision tree. A decision tree processor **124** may include both circuitry to score a decision tree, and the decision tree code itself, embodied as a decision tree table and stored in some memory accessible to the decision tree processor **124**. One or more decision tree tables may be hard-coded into the decision tree processors **124**, stored on memory within the decision tree processors **124**, or stored on memory that is otherwise associated with and communicatively coupled to the decision tree processors **124**. The memory that the decision tree tables are stored in may be shared or dedicated storage, and may be random-access memory (RAM), flash memory, read-only-memory (ROM), or other memory type. The memory that the decision tree tables are stored on may be on-die, such as on-die memory, or may be off-chip on attached memory, such as may be communicatively coupled via a high-speed memory interface. The model contexts may be co-resident within the shared or dedicated memory. In some embodiments the host **104** may provide the model contexts **108** to the decision tree scorers **102**, and/or to an on-chip or attached memory. The host **104**, when scheduling a workload, may provide the decision tree scorers **102** an indication of the model context **108** that should be loaded or otherwise accessed and executed by the decision tree processors **124**. In some embodiments, there may be two levels of memory that stores decision tree tables; a first level of memory (which may be on-chip or attached memory, and may be shared or dedicated to one or more decision tree processors **124**) is loaded or loadable with a particular decision tree table or tables to be executed according to a current workload requirement. A second level of memory (which may be on-chip or in attached memory, shared or dedicated to one or more decision tree processors **124**) may store one or more co-resident model contexts, all or portions of which are loadable onto the first level of decision tree table memory.

[0043] The host **104** may provide a common one of the feature vectors **112** to a plurality of specialized or programmable logic devices **116**, and also provide decision tree tables of a single model context **108** to the plurality of specialized or programmable logic devices **116**. Thus, the individual decision tree clusters **122** and decision tree processors **124** across a plurality of specialized or programmable logic devices **116** may score decision trees of a single model context **108** against a common one of the feature vectors **112**. Score data from each of the plurality of specialized or programmable logic devices **116** may be propagated within each of the plurality of specialized or programmable logic devices **116** as described elsewhere within this Detailed Description, and also passed back to the host **104**. In some embodiments, score data may be passed from a first specialized or programmable logic device **116** to another specialized or programmable logic device **116**, which may then further propagate the score data (such as by summing or appending scores, or appending sums of scores) to produce combined score data for both specialized or programmable logic devices **116**.

[0044] Other methods of processing score data are possible without departing from the scope of embodiments. For example, each decision tree scorer **102** may receive scores, or a list of sums of scores, from the decision tree processors **124** and/or the decision tree clusters **122** within the decision tree scorer **102**, and provide a final summed value either to the host **104**, another programmable logic device **116**, or to some other downstream device. The decision tree scorer **102** may provide the lists of scores (or sums of scores) to the host **104**, another programmable logic device **116**, or to the other downstream device. The host **104**, other programmable logic device **116**, or other downstream device may perform a final scoring of the feature vector **112**, such as by summing the scores or performing some other algorithm to determine a final score for the feature vector **112**, such as based on score data from one or more of the of the plurality of specialized or programmable logic devices **116**.

[0045] In some embodiments, the specialized or programmable logic devices **116** may be, or be included in, one or more of application-specific integrated circuits (ASIC), a programmable logic device such as a field programmable gate array (FPGA), a system on a chip (SoC), as part of a general purpose processor having a specialized portion that scores decision trees, some other logic device, or some combination of the above.

General Purpose Processor with Extended Instruction Set

[0046] In some embodiments, the instruction set architecture of a general purpose processor is extended to include decision tree traversal, scoring instructions, and state. In some embodiments, the extended instruction set includes an instruction to walk one node in a decision tree. In some embodiments, the extended instruction set includes an instruction to walk a plurality of nodes, or to walk an entire decision tree from a root (top node) to a leaf.

[0047] The state usable by a general purpose processor with an extended instruction set to traverse a decision tree includes representation of the decision tree nodes and the feature vector data. The decision tree nodes may be represented in a data structure, in executable instructions, or in some other form. As a data structure, the decision tree may be represented as a tree comprising one or more nodes, the nodes comprising feature identifiers, threshold values, and left and right subtree data, which may identify left (respectively right) subtree nodes or left (respectively right) leaf nodes or leaf score values. A particular node's data may be bundled into adjacent bytes e.g. a record or 'struct' or 'class', or may be spread across tables. Where the decision tree nodes are represented as a data structure, a tree node is identified by a data value, e.g., an index or pointer (machine address) of the node. Traversing a tree node responsive to an instruction to walk one or more nodes comprises starting with a tree node identifier, retrieving the feature it identifies, comparing it to the threshold value of the node, and using the comparison outcome to determine the tree node identifier of the left or right subtree, or right or left leaf/leaf value. In some embodiments an instruction to walk a node, referred to herein as a NODEWALK instruction, may take two parameters, for example a register containing a pointer to the tree node and a register containing a pointer to the feature vector in RAM, and may produce two values, for example, a register containing either a pointer to the left or right subtree (if not a leaf node) or containing the output value (if a leaf node), as well as a condition code register containing a flag that is set if NODEWALK has reached a leaf value (terminating the tree walk). In assembly language, a tree walk includes:

```
; load r1 with the address of the root node of the decision tree
; load r2 with the address of the feature vector
repeat:
        r1 = NODEWALK r1,r2 ; walk from one node to its left or right
        subtree
                ;node
        JNE repeat              ; repeat until a leaf is reached
        ; reached a leaf; leaf output value is in r1
```

[0048] Another embodiment of NODEWALK bundles the loop test and jump into one instruction:

```
; load r1 with the address of the root node of the decision tree
; load r2 with the address of the feature vector
repeat2:
        r1 = NODEWALKREPEAT r1,r2,repeat2 ; walk one node,
        repeat ;until
                ;a leaf is reached
        ; reached a leaf, leaf output value is in r1
```

[0049] Another embodiment walks the entire tree in one instruction:

```
; load r1 with the address of the root node of the decision tree
; load r2 with the address of the feature vector
        r3 = TREEWALK r1,r2; walk the tree
        ; leaf output value is in r1
```

[0050] In some embodiments, a decision tree is represented as a series of tree traversal instructions that are executed by a processor, which implements decision tree traversal instructions. The instructions correspond to one node in a decision tree. The instructions, represented as bit strings, comprise bit fields including a feature identifier, a threshold, and identifiers of the left and right subtree nodes and/or leaves and leaf values. In this embodiment, a tree node is identified with an instruction (machine code) address. Therefore a tree walk comprises executing a tree node walk instruction that changes program control flow to jump to the code for the left or right subtree.

[0051] For example, if a decision tree is comprised of two nodes:

```
if (feature 10 <= t1) then
    if (feature 20 <= t2) then
        score = a;
    else
        score = b;
    endif
else
    score = c;
endif
```

[0052] This might be represented by this program:

[0053] treewalk:

```
; load r2 with the address of the feature vector
root:
  r1 = NODE #10,#t1,#left,#0,#0,#c,#end,r2
left:
  r1 = NODE #20,#t2,#0,#0,#a,#b,#end,r2
end:
  ; leaf output value in r1
```

in which the NODE instructions encode:

[0054] leaf-output-value=NODE #feature-identifier, #threshold-value, #left-subtree-address, #right-subtree-address, #left-leaf-output-value, #right-leaf-output-value, #leaf-code-address, feature-vector-address-register

The # fields are 'immediate' value bit fields of the instruction.

[0055] In this embodiment a decision tree is scored by executing its first NODE instruction, which jumps to the next left or right NODE instruction, and so on, until it reaches a leaf. The root NODE instruction's bit fields encode the feature identifier (10), the threshold value (41), the left subtree ('left'), the right subtree (nil), the left leaf value (nil), and the right leaf value (c). In this example if the identified feature is less than or equal to the threshold t1, then control transfers to the second NODE instruction at address 'left'. This instruction's bit fields encode its feature identifier (20), threshold value (t2), left and right subtrees (nil), and the left and right leaf output values (a and b, respectively). If a node instruction advances to a leaf, then it transfers control to the specified address (end) and the leaf index or output value is obtained in the output register.

[0056] In other embodiments, a tree traversal instruction may use implicit registers, special purpose registers, or memory locations to identify the feature vector and the leaf-node address. Other embodiments may employ variable-length instruction encodings to compress or eliminate instruction bit fields (such as nil subtree fields) which are not used to represent a particular decision tree node. Other embodiments may take advantage of adjacency of instructions in memory to compress or eliminate bit fields (such as a left or right subtree address) in a manner similar to that described earlier.

[0057] In some embodiments, the decision tree data structure memory, or the decision tree instruction memory, may be integrated into the general purpose processor, stored externally to it, or may be coupled to external memory through a memory cache hierarchy.

[0058] A general purpose processor with decision tree traversal, scoring instructions, and state may also be coupled to a feature storage RAM. In some embodiments the feature storage RAM may be loaded automatically by a feature vector distribution network as described elsewhere within this Detailed Disclosure. In particular, new feature data may be loaded into this RAM by the feature vector distribution network without requiring execution of any instructions by the general purpose processor with extended instruction set for walking decision trees. This may save time and energy required to score a decision tree.

[0059] A general purpose processor with decision tree traversal, scoring instructions, and state may also be coupled to a score aggregation system. This may comprise additional registers, thread state, or an adder tree network, to accumulate leaf output values (scores) resulting from instructions like NODEWALK, TREEWALK, or NODE to traverse a node to a leaf node. This too may save time and energy required to score a decision tree.

Example Decision Tree Coding

[0060] FIG. **2** illustrates decision tree coding in accordance with various embodiments of the present disclosure. An example decision tree **200** is illustrated in FIG. **2**. It includes a plurality of decision nodes **202** and a plurality of leaf nodes **204**. A decision node **202** includes various features, including a feature identifier, which may be an address, an index number, a reference numeral or other identifier that identifies the feature being compared at the decision node **202**. The decision node **202** also includes a threshold value to which the feature value (referenced via the feature identifier) is compared. The decision node **202** also includes a left branch pointer and a right branch pointer, which indicate the locations where the next nodes are located. Each decision node **202** represents a comparison; for example node number 7 shows that feature value, identified as feature "F1," is compared to a threshold number 10. Other comparisons are possible without departing from the scope of embodiments.

[0061] Embodiments described herein refer to left branch, right branch, left nodes, right nodes, etc. But these terms are used merely for the sake of describing a decision tree. In general, a decision tree walking algorithm performs a comparison between the feature value and the threshold value and proceeds to either a first node or a second node depending on the outcome of the comparison. For ease of description, these next nodes are referred to herein as left nodes and right nodes, but this is not to be taken in a literal or limiting sense.

[0062] A leaf node **204** includes a leaf value. When a decision tree walking algorithm reaches a leaf node **204**, the particular instance of walking the decision tree is complete, and the leaf value corresponding to the particular leaf node **204** arrived at is output.

[0063] The decision tree coder **106** codes the decision tree **200**. The decision tree coder **106** creates a decision tree table **206** for each decision tree within a model context. In the decision tree table **206**, at least some branch pointers are eliminated with adjacencies. Thus, Node **1** in the decision tree **200** is coded in the decision tree table as being prior to Node **2**. Node **3** is listed after Node **2**, and Node **4** is after Node **3**. Thus, during the execution of Nodes **1**-**3** within the decision tree table **206**, a decision tree processor, such as one of the decision tree processors **124**, knows to select, based on the outcome of a comparison of the feature value to the threshold value, either the following adjacent node in the decision tree table **206** or another node, referred to by next node data such as an offset value, as a next node to be executed by the decision tree processors. Thus, based on the example adjacencies illustrated in FIG. **2**, the outcomes of executing decision nodes of the decision tree table **206** that indicate to select the left branch result in selecting the adjacent node as the next node. Thus, where a particular decision node has a left branch that leads to another decision node (and not to a leaf node), the adjacent node in the decision tree table **206** is the next left node. Right next nodes are identified using next node data, such as offset values. Where there is no left decision node (because for example the left branch leads to a leaf node), it is possible for right next nodes to be adjacent; such right nodes may also identified by next node data, such as offset values, or they may be assumed to be adjacent.

[0064] In addition to arranging the decision nodes **202** within the decision tree table **206** based on adjacencies, the decision tree coder **106** also includes any leaf node values of leaf nodes **204** in the representation of the decision nodes **202** within the decision tree table **206**. For example, Node **7** is coded by the decision tree coder **106** such that its representation includes a leaf value. Based on the outcome of the execution of Node **7** (e.g., based on the comparison of the feature value to a threshold value 10 as shown in FIG. **2**), the decision tree processor selects either to output the value of the left leaf node or select node **8** as the next decision node for processing.

[0065] The decision nodes **202** are represented within the decision tree table **206** as variable length decision nodes (some are shown as being smaller than others to illustrate this). In one example, the following fields are used by the decision tree coder **106** to code the decision nodes.

[0066] 2 Leaves: {feat_ad; info; L_Leaf_val; R_Leaf_val} (72 bits)

[0067] 1 Leaf: {feat_ad; info; L_Leaf_val or R_Leaf_val} (48 bits)

[0068] 0 Leaves: {feat_ad; info; delta(optional)} (24 or 36 bits)

[0069] All representations of decision nodes **202** within the decision tree table **206** include a feature identifier ("feat_ad") and information ("info") field. The feature identifier identifies a location within the feature storage where the feature value (which may be a feature index value as described elsewhere within this Detailed Description) to be compared to a threshold in the execution of the decision node is found. The feature identifier may be an address or other identifier that a decision tree processor uses to look up the feature value within feature storage, such as within the feature storage **126**. The information field includes various sub-fields discussed in more detail below.

[0070] The two-leaf decision nodes also include a left leaf value ("L_Leaf_val") and a right leaf value ("R_Leaf_val"). These values represent possible outcomes or outputs of the decision tree **200**. The one-leaf decision nodes include one of a left leaf value ("L_Leaf_val") or a right leaf value ("R_Leaf_val"). A leaf value may include various data types, including integer, fixed point, floating point, or an index that identifies a unique value stored outside of the decision tree table.

[0071] A decision node with no leaves, such as Node **2**, includes an optional delta value that identifies where the right decision node is located. In this case, the left decision node is located within the decision tree table **206** at the adjacent location (e.g., for Node **2**, the "left" decision node is Node **3**). The right decision node is located at a location within the decision tree table **206** that is identifiable by the delta value. The decision tree processor processes the delta value to determine the right decision node value. For example, the decision tree processor may add the delta value to a location value (e.g., an index value or address) of the currently executing decision node to obtain the location value (e.g., address or index value) of the next right decision node. In some instances, the delta value is included within the info field as described in more detail below. In these instances, a separate delta value is not included within the node representation.

[0072] In an example implementation, the feat_ad field is 12 bits, the info field is 12 bits, the rdelta field is 12 bits, the left leaf value is 24 bits, and the right leaf value is 24 bits.

[0073] The info field includes various sub-fields that identify the threshold value, whether there is a left leaf, whether there is a right leaf, and encodes common offset or delta values for locating the next right node. One example of the info field is as follows:

[0074] Info: {nyb; x; l_leaf; r_leaf; threshold}

[0075] In some embodiments, the nyb field is 1-bit that identifies whether the feature value is a 4-bit or an 8-bit word (e.g., whether the feature value is a "nibble"), the x field is 1-bit, the l_leaf is 1-bit, the r_leaf is 1-bit, and the threshold is 8 bits, although other field sizes may be used without departing from the scope of embodiments. The l_leaf field indicates whether the node includes a left leaf value; likewise, the r_leaf field indicates whether the node includes a right leaf value. As noted above, the info field can be used to code the right node offset or delta value, thereby eliminating the need for a separate delta field in the node. Where x=1, the l_leaf and r_leaf fields are used to code four common offset values. In a particular example, the l_leaf and r_leaf fields are used to code offsets of 8 words, 12 words, 16 words, and 20 words (where a word=12 bits in this particular example), although other offset values may be coded without departing from the scope of embodiments. Where the offset value cannot be coded with one of the common offset values within the info field—because for example the next right node is not at a location that is one of the common offset values away from the current node—the optional separate offset delta field is used. In some embodiments, multiple decision trees are stored in one decision tree table, with appropriate coding identifying the number of decision trees and/or locations of the one or more decision trees within the decision tree table.

[0076] In some embodiments, the decision tree table **206** also includes a DTT header **208**, which codes various aspects of the decision tree table **206**, such as the number of decision trees contained within the decision tree table **206** and starting locations for one or more decision trees within the decision tree table **206**.

Example Threshold and Feature Compression

[0077] Within a given decision tree, or within a plurality of decision trees such as within a model context **108**, decision nodes include feature identifiers and threshold values. In an example decision tree node execution, a feature value (read from the feature vector at a location indexed by the feature identifier feat_ad) is compared to a threshold value. The comparison may be a determination of whether the feature value is less than or equal to a threshold value. If yes, then the left branch is selected; if no, then the right branch is selected. Other types of comparisons are possible without departing from the scope of embodiments, such as less than, greater than, or greater than or equal to. Although various examples of feature value and threshold value encoding described below assume that the decision tree comparisons include determining whether a feature value is less than or equal to the threshold value, similar coding can be performed for feature values and threshold values based on other types of decision tree comparisons without departing from the scope of embodiments.

[0078] Throughout a plurality of decision trees, a given feature f_i will be referenced in one or more nodes. The nodes that reference a given feature f_i will include one of one or more threshold values tv_i. Thus, within a given model context (e.g., one or more decision trees), and for a particular feature f_i, the decision tree coder **106** determines a list ts_i of threshold values tv_i that feature values fv_i corresponding to a particular feature f_i are compared to. Threshold values not compared to a particular feature f_i are not included in the list for the particular feature f_i (although they will be included in other lists for other features). To code the threshold values tv_i for a particular f_i within a model context, the following procedure is used.

[0079] For each feature f_i, the decision tree coder **106** forms a sorted list ts_i of all unique threshold values tv_i that are included in any node of any of the decision trees of a model context that also reference f_i. FIG. **3** illustrates an example list ts_i of unique threshold values tv_i on a real number line **300**. The sorted list ts_i only includes unique ones of the threshold values tv_i; thus a particular threshold tv_i appears in ts_i only once, even if it is included in multiple nodes that reference f_i within a given model context.

[0080] An example coding **302** for the thresholds values tv_i in ts_i are shown in FIG. **3**. Index numbers tvi_i are assigned to each unique tv_i on the real number line **300** within ts_i in ascending order, such that the smallest tv_i is assigned index 0 and the largest tv_i is assigned an index number equal to one less than the total number of unique tv_i within ts_i. In the example shown in FIG. **3**, threshold value t1 is the smallest tv_i, and is assigned index 0, while threshold value t6 is the largest tv_i and is assigned threshold index value 5. Where the threshold values are large numbers, the assignment of fixed-point integer index values tvi_i reduces the number of bits required to represent the thresholds within the decision tree node representations. For example, the threshold values tv_i may be 32-bit floating point numbers (although other numbers of bits, and other types of variables, may be used without departing from the scope of embodiments), and in the example illustrated in FIG. **3**, as small as a three-bit number may be used to represent the threshold index values tvi_i (although other numbers of bits may be used to represent tvi_i, and the feature index values fvi_i, as is described in more detail below).

[0081] In addition to coding tv_i into tvi_i, the feature vector coder **110** codes the feature values fv_i that correspond to f_i in the feature vectors **112** into feature index values fvi_i such that they are compatible with the coded threshold index values tvi_i. Each feature vector **112** includes a list of feature values fv_i for each f_i. In some embodiments, the feature values fv_i are coded into feature index values fvi_i based on the threshold index values tvi_i, and in particular they are coded such that outcomes of the comparisons of the threshold index values tvi_i to the feature index values fvi_i are equivalent to the outcomes of comparing the threshold values tv_i to the feature values fv_i. By coding feature values this way, outcomes of the execution of the coded decision trees within the model contexts **108** against the coded feature vectors **112** produce the same outputs as conventional, software-based execution of the decision trees based on the original feature values and threshold values.

[0082] Continuing with the example illustrated in FIG. **3**, execution of the decision nodes of the decision trees within the model contexts **108** are based on determining whether a feature value fv_i is less than or equal to the threshold value tv_i. Therefore, in this example, the feature values fv_i are coded into feature index values fvi_i such that

[0083] $fvi_i \le tvi_i$ if-and-only-iff $fv_i \le tv_i$

[0084] More generally, feature values fv_i are coded into feature index values fvi_i such that

[0085] fvi_i compare tvi_i if-and-only-if fv_i compare tv_i

where compare represents the comparison function performed during execution of the decision tree (e.g., one of ≤, ≥, <, or >). In the example shown in FIG. **3**, feature index values fvi_i are selected such that

[0086] fvi_i is the greatest integer such that $fv_i <= ts_i[fvi_i]$,

[0087] or else $fvi_i = \#ts_i$ if $f_i > ts_i[\#ts_i - 1]$.

where $\#ts_i$ is the total number of threshold values t_i associated with a particular feature f_i within a particular model context (e.g., all threshold values tv_i in nodes that reference feature f_i). Stated another way, feature index values fvi_i are selected to be either (1) the threshold index value tvi_i that corresponds to the smallest one of the threshold values tv_i that is greater than or equal to the feature value fv_i, or if all threshold values tv_i are smaller than the feature value fv_i, (2) a number that is greater than the largest threshold index value tvi_i. In the example shown above, the corresponding fvi_i is selected to be a number equal to the total number of threshold values tv_i, which is one larger than the largest tvi_i; however, any number larger than the largest tvi_i may be selected.

[0088] In the example illustrated in FIG. **3**, the feature vector coder **110** codes example features f1-f6 as coding **304**. With respect to feature f1, t2 is the smallest tv_i that is greater than or equal to f1, and thus the feature index value fvi_i for f1 is set to be the same as the tvi_i for t2 (i.e., 1). With respect to feature f6, no tv_i is greater than or equal to f6; thus the fvi_i for f6 is set to a number greater than the largest tvi_i. In the example shown in FIG. **3**, the fvi_i for t6 is set to 6, which is one greater than the largest tvi_i, 5. Also, in the example shown in FIG. **3**, f4 is coded as 3.

[0089] The number of bits selected to code the tv_i and the fv_i associated with a particular fi is, in embodiments, large enough to accommodate $\#ts_i$ (the total number of unique tv_i associated with decision nodes that reference f_i). In some embodiments, one of a set of possible index lengths is used to represent tv_i and fv_i, which reduces the complexity of coding tv_i and fv_i. In one particular example, tv_i and fv_i are coded as either 4-bit words, 8-bit words, or as multiple 8-bit words, although other word lengths may be used without departing from the scope of embodiments. In a particular example, the index word lengths are selected such that

[0090] If $\lg(\#ts_i) < 4$, recode tv_i and fv_i into 4 bits, where $\lg(x)$ is the logarithm of x to the base-2.

[0091] Else if $\lg(\#ts_i) < 8$, recode tv_i and fv_i into 8 bits

[0092] Else recode any tv_i and fv_i with $\#ts_i > 255$ threshold comparisons as $(\#ts_i)/255$ separate f_i

Where f_i is recoded into $(\#ts_i)/255$ separate f_i, the decision nodes are recoded by the decision tree coder **106** to indicate one of the $(\#ts_i)/255$ separate f_i, and the corresponding threshold values tv_i of the nodes are recoded accordingly. In a specific example, an f_i with 1259 total tv_i within ts_i results in nodes associated with the particular f_i being recoded into one of five different nodes, each with a separate f_i and 8-bit thresholds. Thus, $fvi_i=0$ is coded as (0, 0, 0, 0, 0) (e.g., is coded as 0 for all of the separate f_i that the original f_i is broken into); $fvi_i=255$ is coded as (255, 0, 0, 0, 0) (e.g., 255 for the first of the separate f_i and 0 for all others of the separate f_i); $fvi_i=256$ is coded as (255, 1, 0, 0, 0) (e.g., 255 for the first of the separate f_i, 1 for the second separate f_i, and 0 for all others); $fvi_i=1258$ is coded as (255, 255, 255, 255, 238). Threshold values tv_i for the separate f_i are also coded in a similar way.

Parallel Architecture

[0093] FIG. **4** illustrates architecture **400** of the decision tree scorer **102** implemented on a specialized integrated circuit or a programmable integrated circuit in accordance with various embodiments. The architecture **400** includes a plurality of decision tree clusters (DTC) **122** arranged in a grid. The DTCs **122** are configured to receive model contexts **108** and feature vectors **112** from the decision tree scorer **102**. The DTCs **122** include subsets of a plurality of decision tree processors **124** and subsets of feature storage **126**. The subsets of decision tree processors **124** may be loaded and/or loadable with the same or different decision tree tables as other subsets of the decision tree processors, and the subsets of the feature storage **126** may be loaded or loadable with the same or different feature vectors (e.g., they may be loaded with common feature vectors).

[0094] The DTCs **122** may receive the feature vectors **112** from first neighboring DTCs **122** and distribute them to second neighboring ones of the DTCs **122**. In one example, DTC **122**-A is configured to receive feature vectors **112** from DTC **122**-C, and to distribute those feature vectors **112** to DTCs **122**-C and **122**-D as illustrated by the arrows in FIG. **4**.

[0095] Likewise, the DTCs **122** may receive score data from first neighboring DTCs **122** and propagate them to second neighboring ones of the DTCs **122**. The score data may be based on individual decision tree scores, as output by different ones of the decision tree processors **124** (such as against a common feature set). The score data may a list of scores, a sum of the scores, or some other score data that is based on the individual scores (such as a multiplication of the individual scores, or some other algorithm for processing scores). The lists of scores may be lists of scores from individual decision tree outcomes, or lists of processed scores. For example, all scores from a particular decision tree cluster **122** may be summed, and appended to a list of all DTC **122** scores such that a final score data includes a list of summed scores from each DTC **122**. In another example, all scores from decision trees executed by a single decision tree processor **124** may be summed, and the summed scores from all decision tree processors may be listed in a final score data, and so forth. Other ways of propagating the score data may be used without departing from the scope of embodiments. In some embodiments, processed or raw score data from each DTC **122**, groups of DTCs **122**, decision tree processors **124**, groups of decision tree processors **124**, individual decision trees, group of decision trees are provided to the DTS **400** in some other fashion (such as on a separate output network), and not propagated to neighboring DTCs **122** as described herein.

[0096] In the example shown in FIG. **4**, DTC **122**-E is configured to receive score data from neighboring DTCs **122**-F and **122**-G. The DTC **122**-E is configured to receive score data from neighboring DTCs **122**-F and **122**-G along with score data provided by the decision tree processors (such as the decision tree processors **124**) within DTC **122**-E, process the score data to determine combined score data (such as by summing the scores, appending the scores to a list of individual scores, or processing the score data in some other way), and to pass the combined score data to neighboring DTC **122**-H, which performs similar functions, and so on until all scores are propagated to a final one of the DTCs **122**, which passes the final score data to the DTS **102**. More generally, the DTCs **122** are configured to propagate score data such that scores are not double counted. For example, a particular pattern of score propagation through the DTS **102** avoids any one of the DTCs **122** from receiving two scores from two neighboring DTCs **122** into which the same scores have been processed.

[0097] In some embodiments, loading a model context into the decision tree scorer architecture **400** includes loading different decision tree tables into different ones of the decision tree tiles within the DTCs **122**, including a plurality of decision trees distributed throughout the decision tree processors of the DTCs **122** of the decision tree scorer architecture **400**. In these embodiments, each of the decision trees loaded into the DTCs **122** produces a separate score based on a common feature vector.

[0098] In some embodiments, different decision tree tables loaded at the same time into the decision tree architecture **400** may be part of a single model context **108**, or part of different model contexts **108**. In some embodiments, multiple decision tree models are coded into a single model context. In one example, two models may be similar but have some differences. The decision trees for the two models are modified slightly to introduce new decision nodes that select either model **1** or model **2**. In addition, appropriate features into the feature vectors to select for either model **1** or model **2**.

[0099] In some embodiments, loading a feature vector into the decision tree scorer architecture **400** includes loading the same feature vector values into each of the feature storage tiles of the DTCs **122**. Thus, the plurality of decision trees of the DTCs, which in embodiments are different from one another, are scored against the same set of features, with all scores processed (e.g., summed) and propagated back to the DTS **102**.

[0100] In other embodiments, various ones of the DTCs **122** are loaded with the same decision trees, such that they execute the same decision trees as other ones of the DTCs **122**. Different feature vectors may be loaded into different ones of the DTCs such that the decision trees are executed against different feature vectors. In some embodiments, the DTCs **122** are loaded with different feature vectors and the same decision tree, or group of decision trees, are loaded into the decision tree scorer architecture **400**. In these embodiments, each DTC **122** is loaded with a different group of one or more feature vectors. The decision trees are scored against the feature vectors and scores are accumulated over time for the feature vectors as all decision trees of a model context are flowed past the feature vectors and executed. In these embodiments, the DTCs **122** may be configured to hold scores for the feature vectors until all decision trees of the model context are loaded and executed against the feature vectors; alternatively, individual decision tree scores are transmitted to the host **104**, which accumulates and processes scores for a particular feature vector.

[0101] In still other embodiments, different groups of the DTCs **122** are loaded with different decision tree jobs (e.g., combinations of model contexts and feature vectors). Thus, a first portion of the decision tree scorer architecture **400** determines scores for a first feature vector against a first model context, a second portion of the decision tree scorer architecture **400** determines a score for a second feature vector against a second model context, and so on with an Nth portion of the decision tree scorer architecture **400** determining a score for an Nth feature vector against an Nth model context. In these embodiments, the DTCs **122** of each portion are loaded with decision trees of a model context, and feature vectors distributed one-by-one within the portions for scoring, or the DTCs **122** of each portion are loaded with different feature vectors, and the decision trees of the model context are distributed one-by-one within the portions for scoring.

[0102] The number of DTCs **122** within the decision tree scorer architecture **400** can scale up to an arbitrarily large number, depending on the size and capabilities of the integrated circuit onto which the decision tree scorer architecture **400** is implemented.

[0103] In some embodiments, more than one decision tree scorer architecture **400** is utilized, each with its own set of DTCs **122** executing in parallel. In these embodiments, a single model context may be loaded onto DTCs **122** of one or more chips, and feature vectors distributed to the DTCs **122** of the different chips one-by-one for scoring. In other embodiments, different feature vectors are loaded into the DTCs **122** of the different chips, with different decision trees of the model contexts distributed one-by-one into each of the DTCs **122** for scoring. In various other embodiments, combinations of these approaches may be utilized for different portions of the combined multi-chip decision tree scorer architecture **400**.

[0104] In some embodiments, determining an overall or combined score for the model context loaded into the decision tree scorer architecture **400** is based on an associative function, such as addition or multiplication, where the order in which the scores are grouped is not determinative of the outcome. Thus, the distribution of the decision trees within ones of the DTCs **122** is not necessarily important to producing the correct final or combined score for a particular feature vector against the decision trees of the model context loaded into the architecture **400**. In other embodiments, processing of the scores for a feature vector and model context decision tree scoring job is not associative, and an order in which the decision trees and/or feature vectors are distributed throughout the architecture is important for determining the final or combined score for a particular feature vector.

[0105] Feature vectors **112**, decision tree tables of a model context **108**, and/or score data may be distributed to DTCs **122** and/or decision tree processors **124** via one or more networks, internal to the specialized or programmable logic devices **116**. One or more of the DTCs **122**, the decision tree processors **124**, and the various feature storages **126** may be addressable via packet headers. Regardless of the distribution method for decision tree tables that are loadable into shared or dedicated storage for the decision tree processors **124**, the decision tree tables may be individually transmitted (such as via packets) and addressed to ones of the DTCs **122** or decision tree processors **124**, or the decision tree tables may be distributed together. Logic within the host **104** and/or the decision tree scorer **102** may determine a distribution of the individual decision tree tables amongst the DTCs **122** and the decision tree processors **124**. Furthermore, the DTCs **122** may include logic to distribute decision tree tables to individual ones of the decision tree processors **124**.

[0106] FIG. **4** illustrates an example of a network to distribute scores and feature vectors to the decision tree processors of the on-chip multi-processor system. In particular, the DTCs **122** act as network elements to aggregate/process the score data and feature vectors. In other embodiments, other network types are employed to distribute the scores and/or the feature vectors to the decision tree processors and/or the feature storage. In these other embodiments, the decision tree clusters **122** may or may not be included as part of the architecture. In one embodiment, the decision processors may be arranged in a mesh of decision tree processors, and scores and/or feature vectors may be distributed via the decision tree processors directly, and eventually to the decision tree scorer or other score aggregation element. In other embodiments, a broadcast network—which may be bus, mesh, point-to-point, hub-and-spoke, or other topology—may connect the decision tree processors (and/or decision tree clusters **122**) to the decision tree scorer or other element that provides the feature vectors and/or receives/accumulates/processes scores from the decision tree processors. In other embodiments, a network on a chip (NOC), which may have other purposes such as to distribute configuration data to FPGA elements or other function, may be re-used to distribute feature vectors and/or provide score data from the decision tree processors to the decision tree scorer or other score aggregation element.

[0107] A score aggregation element may receive and accumulate score data from the decision tree processors and/or the decision tree clusters **122**. The score aggregation element may process the score data, which may include summing the score data, appending the score data to a list or vector of scores, perform some other algorithm to compute a score based on the received data, and so forth. The score aggregation element may pass the score data, either processed or in raw form, to a host or other downstream element.

[0108] Embodiments may include separate networks, one for score data and the other for feature vectors. Thus, in different embodiments, a network may be a feature network, a score aggregation network, or both. In some embodiments, decision tree clusters **122** may act as network elements for one or both the feature network or the score network. Other examples are possible without departing from the scope of embodiments.

[0109] FIG. **5** illustrates architecture **500** of a decision tree cluster **122** implemented on a specialized integrated circuit or a programmable integrated circuit in accordance with various embodiments. The architecture **500** includes one or more decision tree processors **124** and one or more feature storages **126**. The example architecture **500** illustrated in FIG. **5** includes five decision tree processors **124** and one feature storage **126**, although other numbers of decision tree processors **124** and feature storages **126** are used in various other embodiments.

[0110] The DTC **122** includes a feature input bus register to receive feature vectors for storage in the feature storage **126**, for example a 64-bit feature input bus register. The DTC **122** includes a score output register to accumulate and output hold scores for output to neighboring ones of the DTC **122**, for example a 33-bit fixed point score output register. An adder tree of the DTC **122** totals the scores from the decision tree processors **124** and from one or two or more neighboring DTCs **122**. The Decision tree processors **124** output done flags when all decision tree threads being executed therein have completed and output scores. The DTC **122** accumulates the done flags, and upon the adder tree adding the scores from neighboring DTCs **122** to the scores from the decision tree processors **124**, the DTC **122** outputs the scores to one or more neighboring DTCs **122**. At this point, the DTCs **122** also output completion signals to their upstream DTCs **122**, such as through a completion signal network, which may be the same as or different from interconnect networks within the DTS **102** to distribute feature vectors, score data, and/or decision tree table data. In the case of a final DTC **122**, the scores and completion signals are output to the DTS **102**. Upon receiving completion signals, the DTCs **122** and the decision tree scorer **102** determines that the upstream DTCs **122** have completed their decision tree execution and that all available scores are received on an input bus, that no more scores are waiting to be received, and that the scores are ready to be added to scores of the decision tree processors **124** and propagated to downstream DTCs **122** and/or the decision tree scorer **102**.

[0111] The feature storage **126** is, in some embodiments, double-buffered to enable one set of features to be loaded into the feature storage **126** while another set of features is read by the decision tree processors **124**. In one example, the feature storage **126** includes two 32-bit write ports, enabling the feature storage **126** to retire 64 bits of features data at 250 MHz. In one example, the feature storage **126** includes two 32-bit read ports to enable the feature storage **126** to receive two 8-bit features per cycle. The feature storage **126** receives a feature identifier from the decision tree processors **124** and responds with a feature value, for example an 8-bit feature value, and a flag.

[0112] In some embodiments, storage space on the feature storage **126** is reduced by selective capture of subsets of the feature vectors that are used by the decision tree processors **124** of the particular decision tree cluster **122**. Not all features within the feature vectors **112** will be referenced by the decision trees of a particular decision tree cluster **122**; thus, the storage space on the feature storage **126** is reduced, in some embodiments, by only capturing those feature values that are actually referenced by the decision trees executed by decision tree processors **124** of the particular DTC **122**. Thus, the portions of the feature vectors to be stored by a particular feature storage **126** may be referenced in a packet addressed to the feature storage **126**, or to the DTC **122** that the particular feature storage **126** is included in. The feature storage **126** may be provided with a mask, such as in a packet addressed to the feature storage **126** or the DTC **122**, that identifies the portions of the feature vector to selectively store.

[0113] As will be described in more detail below, the decision tree processors **124** are multi-threaded tree-walking engines, capable of executing a plurality of decision trees. The decision trees are stored as decision tree tables within the decision tree processors **124**. In various embodiments, the decision tree tables are stored on various memory storage types, such as random access memory, including Dynamic Random Access Memory (DRAM), Block Random Access Memory (BRAM), Static Random Access Memory (SRAM), and so forth. In some embodiments, the decision tree processors **124** include a five-stage pipeline as is described in more detail below; thus, as long as there are at least five runnable threads (corresponding to five decision trees whose execution have not yet completed), the decision tree processor **124** is able to initiate walking one node of a decision tree on each clock cycle.

Multi-Stage Tree-Walking Pipeline

[0114] In some embodiments, the decision tree processors include a pipelined architecture. FIG. **6** illustrates a multi-stage, multi-threaded, pipelined tree walking circuit **600** of a decision tree processor, in accordance with various embodiments. The circuit **600** is implemented on logic circuitry within the decision tree processor. A thread circuit (or thread stage) (denoted "TH" in FIG. **6**) receives a next thread TH_THD from a NEXT_THDS table **602**. In the example illustrated in FIG. **6**, the NEXT_THDS table **602** is 32×5 bits, and thus stores up to 32 5-bit next thread numbers; therefore up to 32 threads can be handled by the circuit **600**. The NEXT_THDS table **602** is a linked list of threads; initially all threads are listed in the NEXT_THDS table **602**; as threads complete (by outputting a leaf value), the threads are de-linked from the NEXT_THDS table **602**. Once all threads are de-linked from the NEXT_THDS table **602**, the decision tree processor outputs a completion signal to the decision tree cluster, indicating that it is finished with all threads. The thread circuit uses the next thread identifier from the NEXT_THDS table **602** to issue a read for the next node address of the next thread from a node address table, NODE_ADS table **604** and a leaf table, LEAFS table **606**. The NODE_ADS table **604** is 32×13 bits, and thus stores up to 32 13-bit next node addresses, one for each thread.

[0115] The LEAFS table **606** stores leaf output flags; where an entry for a particular thread within the LEAFS table **606** stores an output flag (e.g., a 1 or a 0), the leaf value is output to the decision tree cluster and the thread is de-linked from the NEXT_THDS table **602**.

[0116] Where the leaf output flag indicates that no leaf value is previously selected, the next node addresses are passed to the read node circuit (or read stage) (denoted "RN" in FIG. **6**), and a read to the node table NTAB **608** that corresponds to current thread is issued by the circuit **600** for the next node descriptor. In embodiments, the NTAB **608** is stored on dedicated memory within or otherwise associated with the circuitry of decision tree processor. In other embodiments, the NTAB **608** is stored in a memory that is separate from and communicatively coupled to the decision tree processor. In some embodiments, the NTAB **608** is stored in a memory shared by a plurality of decision tree processors.

[0117] The 12-bit feature address F1_FEAT_AD and 12-bit info field F1_INFO of the node descriptor, along with next node data, such as an offset value if present in the NTAB **608**, are read out in the F1 feature circuit (or F1 feature stage) of the circuit **600**. For example, the next left and right node addresses and next left and right leaf flag values are pre-computed by logic **610** at this stage and are a function of the node address, the info field, and the optional rdelta offset field. The info field determines whether the node one, two, or zero next subtree nodes, and whether there are one, two, or zero leaf values. The next left and right next node addresses are pre-computed based on adjacencies within the NTAB **608** the F1_RDELTA value, if present, or from the coding of the offset value in the info field, as described elsewhere within this Detailed Description. In some embodiments, when the current node has a left subtree node, the next left node address is the address of the node adjacent to (immediately following) the current node, and the next left leaf flag is false. Otherwise the current node has a left leaf output value, and the next left node address the address of the word(s) within the current node that contain the left leaf value, and the next left leaf flag is true. The pre-computation is similar for the next right node address and next right leaf flag. When the current node has a right subtree node but no left subtree node, the next right node address is the address of the node adjacent to (immediately following) the current node, and the next right leaf flag is false. When the current node has both a left subtree node and a right subtree node, the next right node address is determined by adding the current node address and an offset (whose value is encoded within the info field, or explicitly represented in the optional rdelta offset field), and the next right leaf flag is false. Otherwise the current node has a right leaf output value, and the next right node address the address of the word(s) within the current node that contain the right leaf value, and the next right leaf flag is true.

[0118] At the F2 feature circuit (or F2 feature stage) of the circuit **600**, the feature value associated with the F1_FEAT_AD is read from the feature storage **612** (e.g., the feature storage **126**). The FST **126**, in embodiments, is configured to be read by two different decision tree processors; thus the feature storage **612** is shown having two inputs and two outputs.

[0119] At the execution circuit (or execution stage) of the circuit **600** (denoted "EX" in FIG. **6**), the feature value ("EX_FEAT") read from the feature storage **612** is compared by logic **614** to the threshold value (EX_TH) of the currently executing node. The threshold value EX_TH and the feature value EX_FEAT may be threshold index values and feature index values as is described elsewhere within this Detailed Description, or they may be uncompressed threshold values and feature values. Embodiments of the present disclosure are not limited to use of one or the other. Based on the outcome of the compare output by the logic **614**, either a next left node address or a next right node address is written to the thread's entry in the NODE_ADS table **604** Also based on the outcome of the compare output by the logic **614**, either a next left leaf flag or a next right leaf flag is written to the thread's entry in the LEAFS table **606**.

[0120] Once the execution circuit selects a leaf value for a particular thread and sets a leaf flag, then the next time the thread is issued into the pipeline, the leaf flag is read and the node address in the NODE_ADS table **604** is not the address of a node but rather the address of leaf value words within previous node within the NTAB **608**. At the RN circuit, these leaf value words are read from the NTAB **608**, thereby obtaining the leaf value's score **620** for the particular thread instead of a feature address and info field. The score **620** may be output to the decision tree cluster as described elsewhere within this detailed description. In addition, when the leaf flag is true, the thread is unlinked from the NEXT_THDS table **602** so that it is not fetched by the pipeline again.

[0121] Each of the portions of the circuit **600** (TH, RN, F1, F2, and EX) concurrently processes different ones of the threads. Thus, at any one time, the circuit **600** processes some portion of up to five different threads, which corresponds to processing some portion of up to five different decision trees concurrently, every clock cycle.

Example Processes

[0122] FIG. **7** depicts a flow graph that shows an example process **700** of executing a decision tree, in accordance with various embodiments. At **702**, a decision tree processor, e.g., a thread circuit or stage of a decision tree processor pipeline, determines a next thread to be executed by the processor and issues a read to the node table to determine the next node address of the next thread.

[0123] At **704**, a decision tree processor, e.g., a read node circuit or stage of a decision tree processor pipeline, retrieves decision tree node data, such as decision tree node words, including at least feature indicators and threshold values, from a decision tree node table, which may be stored within the decision tree processor. A subset of the decision tree nodes also includes next node data, such as next node offset values.

[0124] Final outcomes of the decision tree node executions result in output of leaf values as an output of the decision tree-walking thread, such as where a decision tree node execution results in selecting a left leaf or a right leaf value. At **706**, the decision tree processor, e.g., a read circuit or stage of a decision tree processor, determines whether a leaf flag is set for a particular thread, such as during a previous pass of the thread through a pipeline. Where the leaf flag is set, at **708** the particular thread is unlinked from the threads table. At **710**, leaf value data, such as one or more leaf value words, of the decision tree node are read by the read node circuit or stage of the decision tree processor pipeline and output to the decision tree cluster, or to some other output network.

[0125] At **712**, where the leaf flag value is not set, the decision tree processor, e.g., a feature circuit or stage of a decision tree processor pipeline, reads the feature value identified by the feature indicator from feature storage.

[0126] At **714**, the decision tree processor, e.g., the feature circuit or stage of a decision tree processor pipeline, pre-computes possible next decision tree node addresses based on the next node data, such as offset values and the next decision tree nodes that are adjacent to currently executing nodes. The decision tree processor, e.g., the feature circuit or stage of a decision tree processor pipeline, also or alternatively pre-computes addresses for right or left leaf data, such as right or left leaf words or values of the current decision tree node. As noted elsewhere within this Detailed Description, a subset of the nodes of the decision tree node table includes one or more leaf values. The presence of leaf nodes indicates that a possible outcome of the execution of the decision node is to select to output a leaf value the next time the thread passes through the pipeline. Thus, the decision tree processor pre-computes one of a left leaf data address or a left next node address, and one of a right leaf data address or a right next node address, depending on whether there is a left leaf or left next node, and based on whether there is a right leaf value or a right next node in the particular decision node being executed. Pre-computation at **714** occurs prior to the execution of the decision node by the decision tree processor.

[0127] Pre-computation of some of the next node addresses is performed, in some embodiments, by processing next node data, such as an offset value of the decision tree node, such as by adding the offset value to location of the current node to arrive at a location of the next node. The next node data, such as an offset value, is either separate next node data, such as an offset value, provided within the decision node, or coded by the info field of the decision node, as described elsewhere within this Detailed Description. Pre-computing the next node addresses is also based on adjacencies for some of the next node addresses.

[0128] At **716**, the decision tree processor, e.g., an execution circuit or stage of the decision tree processor pipeline, executes the decision tree node. Executing the decision tree node includes comparing a threshold value of the decision tree node to the feature value retrieved from the feature storage. The threshold value may be a threshold index value, and the feature value may be a feature index value, as described elsewhere within this Detailed Description.

[0129] At **718**, the decision tree processor, e.g., the execution circuit or stage of the decision tree processor pipeline, determines the next decision tree node for the thread to be retrieved and executed and/or an address of leaf data containing a leaf value to be output the next time the thread is fetched into the pipeline. Selection of the next decision tree node or address of leaf data are determined based on an outcome of executing the decision tree node. Some outcomes of the comparisons (such as where the feature value is less than or equal to the threshold value) result in determining the next decision tree node based on the next node data, such as a next node offset value. Other outcomes of the comparisons (such as where the feature value is not less than or equal to the threshold value) result in determining the next decision tree node that is adjacent to currently executing node within a decision tree table associated with the decision tree within the decision tree processor.

[0130] At **720**, a determination is made by the decision tree processor, e.g., by the thread circuit or stage of the decision tree processor, whether all threads have been retired. As threads output leaf values at **710** and are completed, they are de-linked at **708** from a linked list of decision tree threads. When all threads are de-linked, the decision tree executions in this decision tree processor **124** are complete. Each thread corresponds to a single decision tree; thus once all threads are completed, the decision tree processor outputs a completion signal and outputs one or more scores from the decision tree execution.

[0131] FIG. **8** illustrates a process **800** of scoring a plurality of decision trees by a decision tree scorer, in accordance with various embodiments. At **802**, the decision tree scorer **102** receives a model context **108** from a host **104** or other upstream processing system. At **804**, the decision tree scorer **102** loads the model context **108** onto the plurality of decision tree clusters **122**.

[0132] At **806**, the decision tree scorer **102** receives a feature vector **112** from the host **104** or from an upstream processing system. At **808**, the decision tree scorer **102** provides the feature vector **112** to a first one of the decision tree clusters **122**. Thus, in some embodiments, a common feature vector is provided to the decision tree clusters **122** and the decision tree processors **124**.

[0133] At **810**, the decision tree scorer **102** receives a final score and a completion signal from one of the decision tree clusters **122**, indicating that the decision tree clusters have completed the scoring of feature vector with the plurality of decision trees. At **812**, the decision tree scorer **102** provides the final score to the host **104** or a downstream processing system, which may include in some embodiments another decision tree scorer or other system.

[0134] FIG. **9** illustrates a process **900** of scoring a plurality of decision trees by decision tree clusters, in accordance with various embodiments. At **902**, the decision tree clusters (DTCs) **122** receive a feature vector (such as a common feature vector) from the decision tree scorer **102** or from neighboring DTCs **122**. At **904**, the DTCs **122** provide the feature vector to other neighboring DTCs **122**. In this manner, the feature vector is distributed to all DTCs within a decision tree scorer.

[0135] At **906**, the decision tree clusters **122** cause the plurality of decision tree processors **124** within a plurality of DTCs **122** to begin execution of the plurality of decision trees within the model contexts loaded onto the DTCs **122**. The execution of the plurality of decision trees may be concurrent, and may be performed by multi-threaded, multi-stage pipelined decision tree processors. Execution of the decision trees includes, among other things, comparisons of threshold values (or threshold index values) to feature values (or feature index values) of a common feature vector, and selection of next nodes and/or output values based on the comparisons. The execution of the decision trees results in corresponding scores for ones of the plurality of decision trees.

[0136] At **908**, the DTCs **122** receive from the decision tree processors **124** the corresponding scores and completion signals resulting from the execution of the decision trees on the decision tree processors **124**. At **910**, the DTCs **122** receive scores and completion signals from neighboring DTCs **122**.

[0137] At **912**, based on receipt of the completion signals and the scores, the DTCs **122** process the scores from the decision tree processors **124** within the DTCs **122** with scores from the neighboring DTCs **122**. For example, the DTCs **122** may sum the scores to produce an accumulated score. In another example, the DTCs **122** may append the scores, or a sum of the scores from the decision tree processors **124** within the DTCs **122**, to the score data received from the neighboring DTCs **122**.

[0138] At **914**, the DTCs **122** propagate the accumulated scores and completion signals to neighboring DTCs **122**, eventually reaching the final one of the DTCs **122**, which provides a final score to the decision tree scorer **102**. In this way, the individual scores from each of the decision trees executing on the decision tree processors **124** within each of the DTCs **122** are accumulated into final score data, such as a final sum of scores or list or set of scores from individual ones of the decision tree processors and propagated to the decision tree scorer **102**.

[0139] FIG. **10** illustrates a process **1000** of coding threshold values of a plurality of decision trees in accordance with various embodiments. At **1002**, a decision tree coder **106** identifies all threshold values referenced in all decision nodes of a plurality of decision trees—such as those within a model context **1008**—that correspond to a particular feature.

[0140] At **1004**, the decision tree coder **106** determines a list of unique threshold values associated with the particular feature in the one or more decision trees. In some embodiments, the list is sorted, such as in ascending or descending order. At **1006**, the decision tree coder **106** determines a number of bits to be used to represent threshold index values for the threshold values based at least in part on a number of values in the sorted list of unique threshold values associated with the particular feature in the one or more decision trees.

[0141] In one particular example, where the base-2-logarithm of the total number of threshold values associated with the particular feature is less than 4, the threshold index is coded as a 4-bit word, and where the base-2-logarithm of the total number of threshold values associated with the particular feature is less than 8, the threshold index is coded as a 8-bit word. Where the base-2-logarithm of the total number of threshold values associated with the particular feature is greater than 8, multiple features are used to represent the particular feature in the coded decision tree, such that the number of features to represent the particular feature is determined by n/255, where n is equal to the total number of threshold values associated with the particular feature, as described elsewhere within this Detailed Description. 8-bit words are used to represent the threshold values for these multiple features. In other embodiments, the decision tree is modified to include multiple decision nodes in place of one node with a number of unique thresholds exceeding a predetermined value. Other examples are possible without departing from the scope of embodiments.

[0142] At **1008**, the decision tree coder **106** determines a plurality of threshold index values for the list of unique threshold values. In some embodiments, index values are assigned to the sorted list, such that threshold index values associated with smaller threshold values are smaller than threshold index values associated with larger threshold values, although larger index values are assigned to smaller threshold values in other embodiments. In one particular example, the smallest one of the unique threshold values is assigned a threshold index value of 0, and the largest one is assigned a threshold index value that is equal to the total number of unique threshold values minus one. Other examples are possible without departing from the scope of embodiments.

[0143] At **1010**, the decision tree coder **106** represents the one or more decision trees such that decision nodes of the one or more decision trees associated with the particular feature include the threshold index values. The process **1000** is repeated for each feature referenced in at least one decision node of a plurality of decision trees until all threshold values in the plurality of decision trees are coded with threshold index values.

[0144] FIG. **11** illustrates a process **1100** of coding a vector of feature values, in accordance with various embodiments. As described above with respect to FIG. **10**, threshold values for each feature are coded. Feature values for feature vectors that are to be scored against the set of coded decision trees are coded such that the feature values are compatible with the coded threshold values. At **1102**, a feature vector coder **110** receives a feature vector **112** to be scored by a plurality of decision trees.

[0145] At **1104**, a feature vector coder **110** compares a feature value associated with the particular feature to the threshold values that correspond to the particular feature (e.g., to the list ts_i described above). At **1106**, a determination is made by the feature vector coder **110** as to whether the feature value corresponding to the particular feature in the feature vector is greater than the largest threshold value in the set of threshold values associated in the plurality of decision trees with the particular feature.

[0146] At **1108**, upon determining that the feature value is not larger than the largest threshold value (the "NO" path), the feature vector coder **110** identifies a smallest one of the list of unique threshold values that is greater than or equal to the feature value.

[0147] At **1110**, the feature vector coder **110** codes the feature value to produce a coded feature value (e.g., a feature index value) that is equal to a particular one of the sorted threshold index values that corresponds to the smallest one of the sorted list of unique threshold values.

[0148] At **1112**, upon determining that the feature value is larger than the largest threshold value (the "YES" path), feature vector coder **110** sets the feature index value to be larger than the largest threshold index value. In one particular example, the feature index value is set to be equal to the total number of unique threshold values associated with the feature, but any number larger than the largest threshold index value could be used. In this way, the feature index values are set such that outcomes of comparisons of the threshold index values to corresponding feature index values during decision tree execution (such as by the decision tree processors **124**) are equivalent to outcomes of comparisons of corresponding threshold values to corresponding feature values.

[0149] The operations of the example processes of FIGS. **7-11** are illustrated in individual blocks and summarized with reference to those blocks. The order in which the operations are described is not intended to be construed as a limitation, and any number of the described operations can be combined in any order, separated into sub-operations, and/or performed in parallel to implement the process. Processes according to various embodiments of the present disclosure may include only some or all of the operations depicted in the logical flow graph.

Example Computing System

[0150] FIG. **12** is a block diagram of an example computing system **1200** usable to perform various methods described herein. The computing system **1200** may be configured as any suitable computing device capable of implementing all or part of a decision tree scoring system, such as the host **104**. According to various non-limiting examples, suitable computing devices may include personal computers (PCs), handheld devices, wearable smart devices, smartphones, tablet computers, laptop computers, desktop computers, gaming systems, electronic media players (such as mp3 players and e-book readers), servers, server farms, datacenters, special purpose computers, combinations of these, or any other computing device(s) capable of storing and executing all or part of the decision tree scoring system described herein.

[0151] In one example configuration, the computing system **1200** comprises one or more processors **1202** and memory **1204**. The computing system **1200** may also contain communication connection(s) **1206** that allow communications with various other systems. The computing system **1200** may also include one or more input devices **1208**, such as a keyboard, mouse, pen, voice input device, touch input device, etc., and one or more output devices **1210**, such as a display, speakers, printer, etc. coupled communicatively to the processor(s) **1202** and the memory **1204**.

[0152] The memory **1204** may store program instructions that are loadable and executable on the processor(s) **1202**, as well as data generated during execution of, and/or usable in conjunction with, these programs. In the illustrated example, memory **1204** stores an operating system **1212**, which provides basic system functionality of the computing system **1200** and, among other things, provides for operation of the other programs and program modules of the computing system **1200**.

Computer-Readable Media

[0153] Depending on the configuration and type of computing device used, memory **1204** of the computing system **1200** in FIG. **12** may include volatile memory (such as random access memory (RAM)) and/or non-volatile memory (such as read-only memory (ROM), flash memory, etc.). Memory **1204** may also include additional removable storage and/or non-removable storage including, but not limited to, flash memory, magnetic storage, optical storage, and/or tape storage that may provide non-volatile storage of computer-readable instructions, data structures, program modules, and other data for computing system **1200**.

[0154] Memory **1204** is an example of computer-readable media. Computer-readable media includes at least two types of computer-readable media, namely computer storage media and communications media. Computer storage media includes volatile and non-volatile, removable and non-removable media implemented in any process or technology for storage of information such as computer-readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, phase change memory (PRAM), static random-access memory (SRAM), dynamic random-access memory (DRAM), other types of random-access memory (RAM), read-only memory (ROM), electrically erasable programmable read-only memory (EEPROM), flash memory or other memory technology, compact disk read-only memory (CD-ROM), digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other non-transmission medium that can be used to store information for access by a computing device. In contrast, communication media may embody computer-readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave, or other transmission mechanism. As defined herein, computer storage media does not include communication media.

CONCLUSION

[0155] Although the disclosure uses language that is specific to structural features and/or methodological acts, the invention is not limited to the specific features or acts described. Rather, the specific features and acts are disclosed as illustrative forms of implementing the invention.

What is claimed is:

1. A decision tree scoring hardware system comprising:

a plurality of decision tree processors to access a common feature vector and to execute one or more decision trees with respect to the common feature vector.

2. The decision tree scoring hardware system of claim **1**, wherein a first one of the plurality of decision tree processors further execute a first one of the decision trees with respect to the common feature vector concurrently with execution of a second one of the decision tree with respect to the common feature vector by a second one of the decision tree processors.

3. The decision tree scoring hardware system of claim **1**, further comprising a decision tree cluster comprising a subset of the plurality of decision tree processors and feature storage to store the common feature vector, the feature storage accessible by the subset of the plurality of decision tree processors within the decision tree cluster.

4. The decision tree scoring hardware system of claim **1**, further comprising:

a first decision tree cluster comprising:

a first subset of the plurality of decision tree processors; and

a first feature storage, accessible by the first subset of the plurality of decision tree processors, to store the common feature vector; and

a second decision tree cluster comprising:

a second subset of the plurality of decision tree processors; and

a second feature storage, accessible by the first subset of the plurality of decision tree processors, to store the common feature vector.

5. The decision tree scoring hardware system of claim **1**, further comprising a plurality of decision tree clusters, ones of the decision tree clusters comprising subsets of the decision tree processors, the subsets of the decision tree processors output scores based on outcomes of execution of the one or more decision trees, a first one of the decision tree clusters to accept first scores from a first subset of the decision tree processors within the first one of the decision tree clusters, accept at least second score data from a second one of the plurality of decision tree clusters, process the first scores and the at least the second score data to produce combined score data.

6. The decision tree scoring hardware system of claim **5**, wherein the first one of the decision tree clusters further processes the first scores and the second score data by one of:

appending the first scores, or a sum of the first scores, to the second score data to produce the combined score data; or

summing the first scores with the second score data to produce the combined score data.

7. The decision tree scoring hardware system of claim **1**, wherein ones of the decision tree processors output scores based on outcomes of execution of the one or more decision trees, the decision tree scoring hardware system further comprising:

a score aggregation network to collect the scores output by the decision tree processors.

8. The decision tree scoring hardware system of claim **1**, further comprising one or more of:

a plurality of decision tree clusters comprising subsets of decision tree processors, wherein a first one of the plurality of decision tree cluster is to receive the common set of features from a second one of the decision tree clusters and to propagate the common set of features to a third one of the decision tree clusters; and

a feature network coupled to the decision tree processors to provide the common set of features to the decision tree processors.

9. The decision tree scoring hardware system of claim **1**, further comprising a plurality of multi-threaded decision tree processors that concurrently executes a plurality of decision trees in multi-threaded processes.

10. A method comprising:

providing a common feature vector to a plurality of decision tree processors implemented within an on-chip decision tree scoring system; and

executing, by the plurality of decision tree processors, a plurality of decision trees, by reference to the common feature vector.

11. The method of claim **10**, further comprising concurrently executing, by the plurality of decision tree processors, the plurality of decision trees by reference to the common feature vector.

12. The method of claim **10**, further comprising:

outputting, into a score aggregation network, scores for ones of the plurality of decision trees based on outcomes of executing the plurality of decision trees.

13. The method of claim **10**, wherein the common feature vector is stored on a feature storage, the method further comprising concurrently:

loading another feature vector into the feature storage; and

executing, by at least one of the decision tree processors, at least one of the plurality of decision trees, the executing including reading one or more features of the common feature vector from the feature storage.

14. The method of claim **10**, further comprising propagating the common feature vector throughout a plurality of decision tree clusters, the propagating including receiving the common feature vector from first neighboring decision tree clusters, and passing, by ones of the plurality of decision tree clusters, the feature vector to second neighboring decision tree clusters.

15. A decision tree-walking system comprising:

feature storage that stores a common feature vector; and

a plurality of decision tree processors that accesses the common feature vector from the feature storage and executes a plurality of decision trees by comparing threshold values of the decision trees to feature values within the common feature vector.

16. The decision tree-walking system of claim **15**, wherein the plurality of decision tree processors concurrently execute the plurality of decision trees.

17. The decision tree-walking system of claim **15**, further comprising:

another feature storage that stores the common feature vector; and

another plurality of decision tree processors that accesses the common feature vector from the other feature storage and executes another plurality of decision trees by comparing other threshold values of the other plurality of decision trees to the feature values within the common feature vectors.

18. The decision tree-walking system of claim **15**, wherein the plurality of decision tree processors is a first plurality of decision tree processors, the on-chip decision tree-walking system further comprising a plurality of decision tree clusters, a first one of the decision tree clusters including the first plurality of decision tree processors, the first one the plurality of decision tree clusters:

receives first score data output by the first plurality of decision tree processors;

receives second score data from one or more first neighboring decision tree clusters;

processes the first score data with the second score data to produce third score data; and

passes the third score data to one or more second neighboring decision tree clusters.

19. The decision tree-walking system of claim **15**, concurrently writing another common feature vector to the feature storage with the access of the feature vector by the plurality of decision tree processors.

20. The decision tree-walking system of claim **15**, wherein the feature storage and the plurality of decision tree processors are first feature storage and a first plurality of decision tree processors included in a first programmable logic device, first application-specific integrated circuit, or first on-chip processor with an enhanced instruction set, the plurality of decision trees are a first plurality of decision trees, and the decision tree-walking system further includes:

a second programmable logic device, second application-specific integrated circuit, or second on-chip processor with an enhanced instruction set that includes:

second feature storage to store the common feature vector; and

a second plurality of decision tree processors that accesses the common feature vector from the second feature storage and executes a second plurality of decision trees by comparing second threshold values of the second decision trees to the feature values within the common feature vector;

the second programmable logic device, second application-specific integrated circuit, or second on-chip processor with an enhanced instruction set:

receives first score data from the first programmable logic device, first application-specific integrated circuit, or first on-chip processor with an enhanced instruction set, the first score data resulting from execution of the first plurality of decision trees; and

processes the first score data with second score data resulting from execution of the first plurality of decision trees to produce combined score data for the common feature vector.

* * * * *