DATA ANALYTICS USING DEEP LEARNING

GT 8803 // FALL 2018 // VENKATA KISHORE PATCHA

Lecture#16: In-RDBMS Hardware Acceleration of Advanced Analytics

CREATING THE NEXT®

TODAY'S PAPER

- In-**RDBMS** *Hardware Acceleration* of Advanced **Analytics**
  - **Authors:**
    - Divya Mahajan, Joon Kyung Kim, Jacob Sacks
      - affiliated with Georgia Tech
    - Adel Ardalan
      - Affiliated with The University of Wisconsin
    - Arun Kumar, Hadi Esmaeilzadeh
      - Affiliated with university of California
  - **Areas of focus:**
    - Data Base; ML, Hardware Acceleration.
  - Slides based on a presentation by Divya @ PVLDB 2018 *

TODAY'S AGENDA

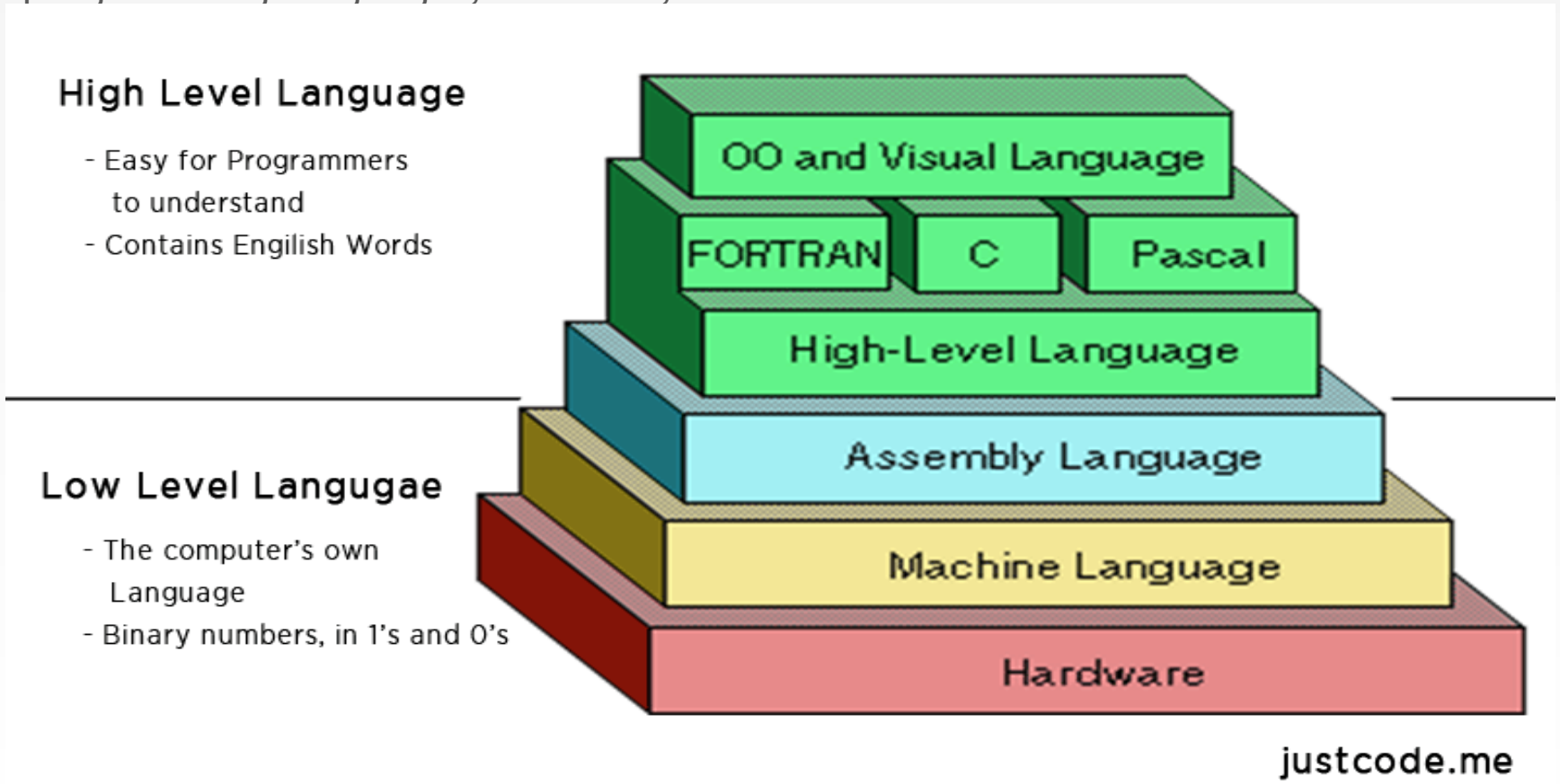
- Background
- Existing work
- Objectives
- Approach
- Experiment
- Resources

BACKGROUND

- CPU cores are powerful, efficient and supports large list of instructions. Today's state of art CPUs have around 10 cores per CPU. CPUs are used by user program through several abstractions. CPUs are supporting extremely large number of application through the support of large list of instructions and software abstractions. They are developed for 'generic' use.

BACKGROUND

- CPUs are used by user program through several abstractions. Application frameworks, multiple programming languages, continers, vertual environments and so on.



BACKGROUND

- What is hardware acceleration?
- If you use any non- CPU hardware that can speed up your program, that is hardware acceleration.

Examples:

Applications	Hardware accelerator
Computer graphics	GPUs are good with 'some' operations but can have thousands of cores in a single GPU. Enables parallel processing. GPUs need CPU to control them.
Digital signal processing	Digital signal processor
Analog signal processing	Analog signal processing
…..	…….
Any computing task	Field-programmable gate arrays (FPGA)

BACKGROUND

- Field-programmable gate arrays (FPGA)
  - FPGA is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL).
  - Example HDLs: VHDL, Verilog.

```
library iEEE; use iEEE.STD_LOGIC_1164.ALL; use
    iEEE.STD_NUMERIC_STD.ALL; entity not1 is port(a:in STD_LOGIC;
    b:out STD_logic); end not1; architecture behavioral of not1 is begin b
    <= not a; end behavioral;
```

https://youtu.be/L2wsockKwPQ?t=15

BACKGROUND

- For a high level language programmers, FPGA do sound cool but not HDL.
  - Luckily, There are many C look a like, python look alike HDL interfaces! MyHDL is python look a like interface that generates HDL.

BACKGROUND

- MyHDL code:

```
def bin2gray(B, G, width):

    """ Gray encoder.

    B -- input intbv signal, binary encoded
    G -- output intbv signal, gray encoded
    width -- bit width

    """

    @always_comb
    def logic():
        Bext = intbv(0)[width+1:]
        Bext[:] = B
        for i in range(width):
            G.next[i] = Bext[i+1] ^ Bext[i]

    return logic
```

- Verilog code:

```
module bin2gray (
    B,
    G
);

input [7:0] B;
output [7:0] G;
reg [7:0] G;

always @(B) begin: BIN2GRAY_LOGIC
    integer i;
    reg [9-1:0] Bext;
    Bext = 9'h0;
    Bext = B;
    for (i=0; i<8; i=i+1) begin
        G[i] <= (Bext[(i + 1)] ^ Bext[i]);
    end
end

endmodule
```

BACKGROUND

- Still complex!
- There are Data base implementations that use FPGA under the hood. User still write only sql queries and care only about their application not signals.
  - doppioDB - A hardware accelerated database
  - Even Postgres, orcale have roadmap or 3rd party plugins that support FPGA.
- Centaur: A framework for hybrid cpu-fpga databases. Centaur is a framework for developing applications on CPU-FPGA shared memory platform, bridging the gap between the application software and accelerators on the FPGA.

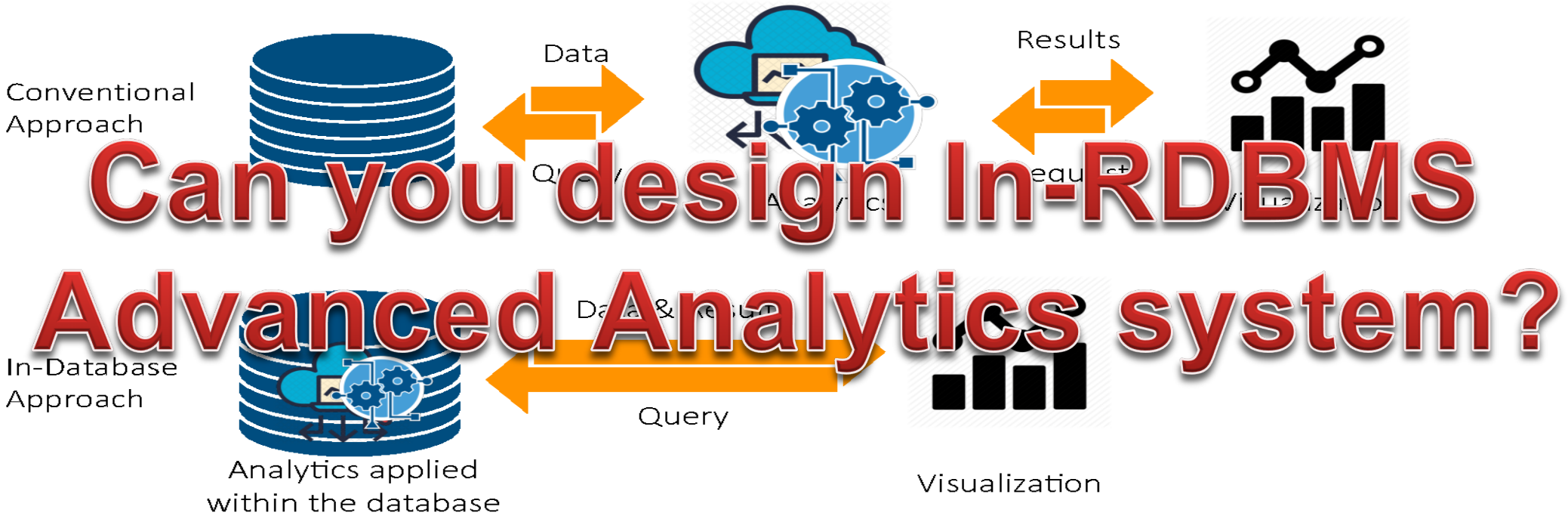
BACKGROUND

- Select pymax(a,b) from ab_table

```
CREATE FUNCTION pymax (a integer, b integer)
  RETURNS integer
AS $$
  if a > b:
    return a
  return b
$$ LANGUAGE plpythonu;
```

- And there is Apache Madlib with all the functions that you need for analytics. Apache Madlib can be deployed to postgres and other Relational databases.

In-RDBMS Advanced Analytics

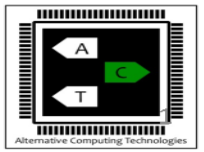


Can you design In-RDBMS Advanced Analytics system?

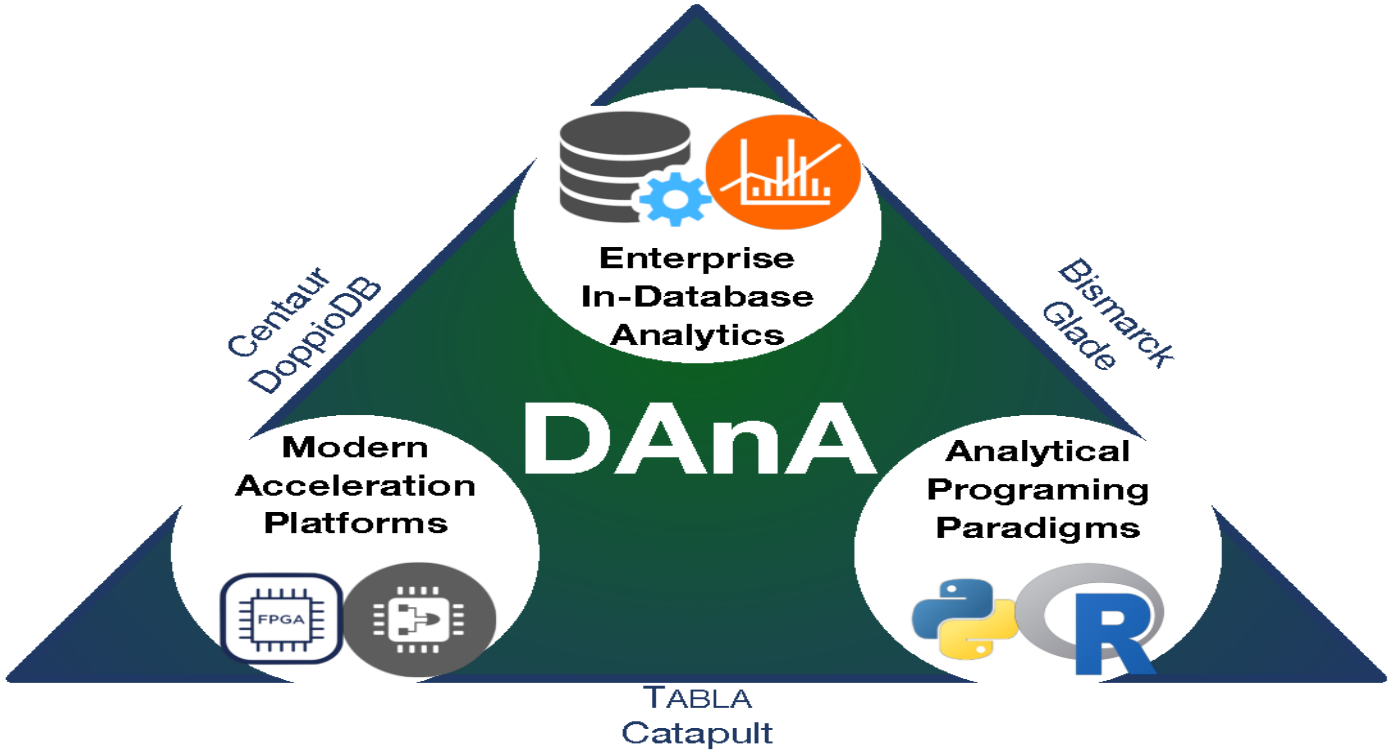
In-RDBMS Hardware Acceleration of Advanced Analytics

Divya Mahajan
Joon Kyung Kim
Jacob Sacks
Adel Ardalan*
Arun Kumar†
Hadi Esmaeilzadeh†

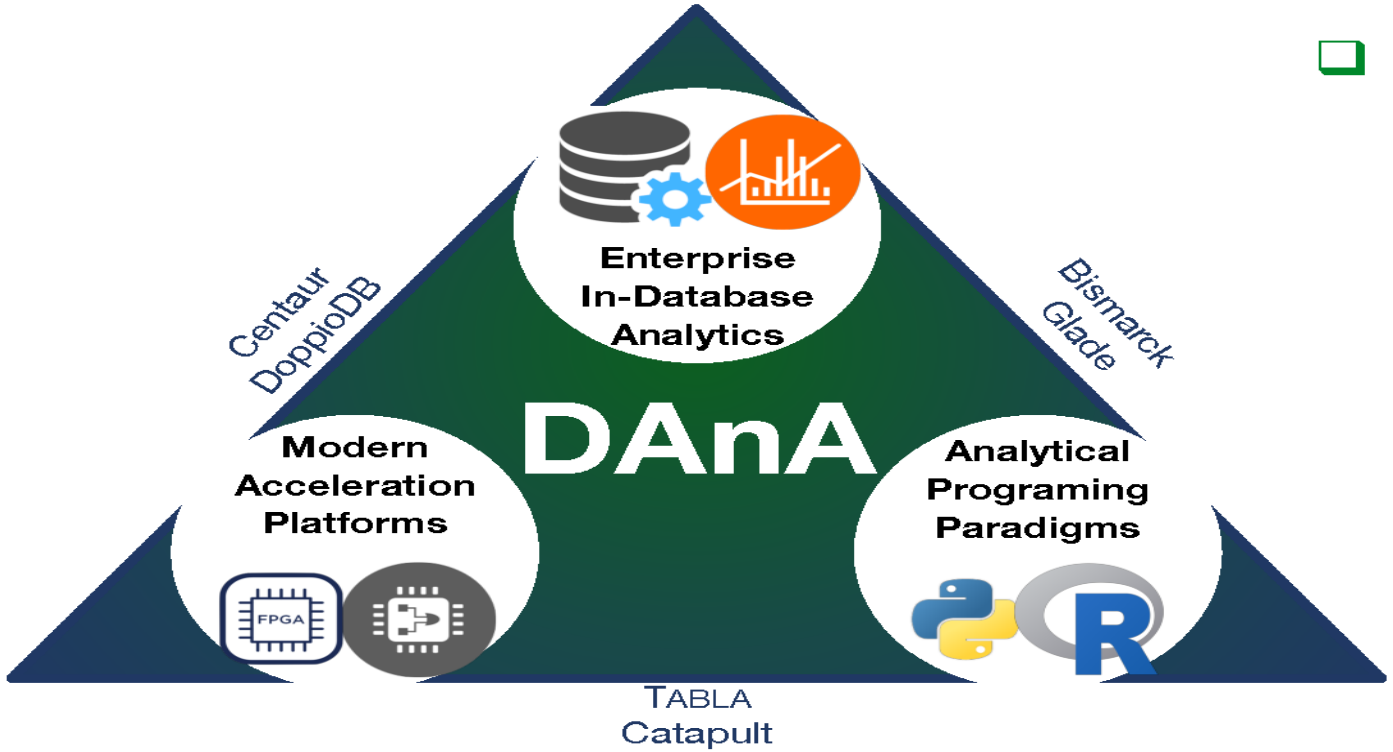
Georgia Institute of Technology
*University of Wisconsin-Madison
† University of California, San Diego
Alternative Computing Technologies (ACT) Lab



DAnA - in-Database Acceleration of Advanced Analytics

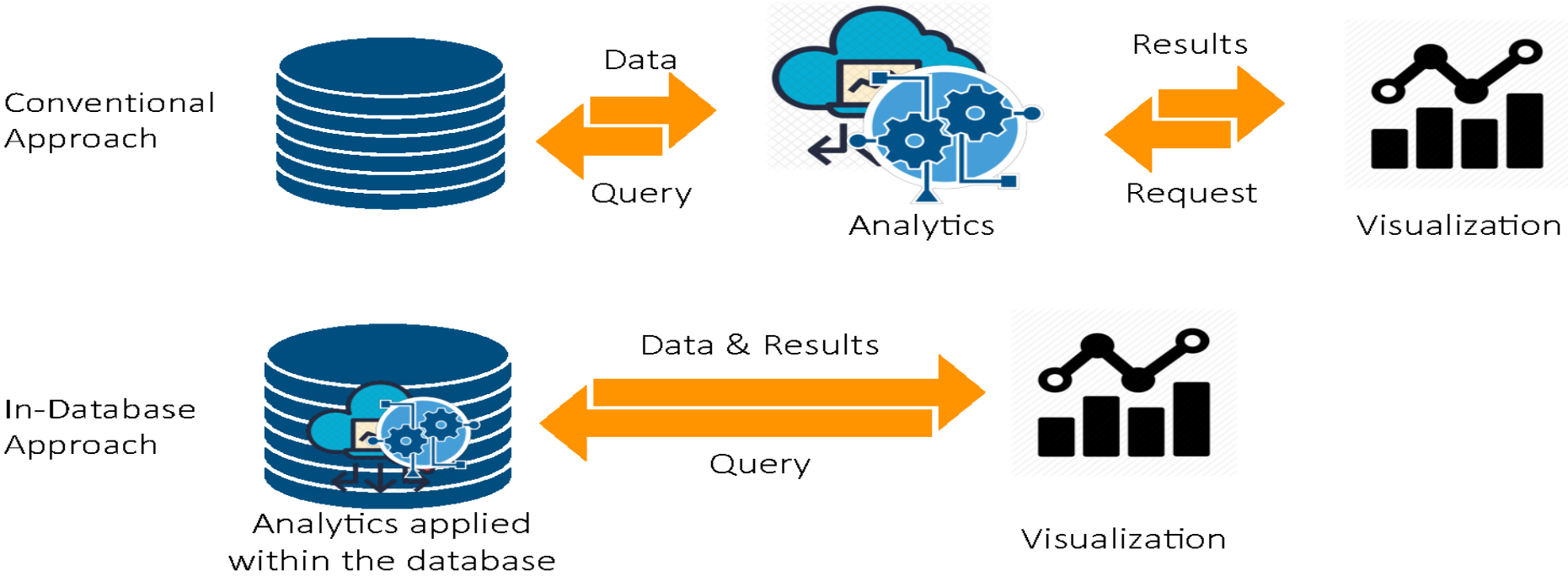


Objectives of **DAnA**

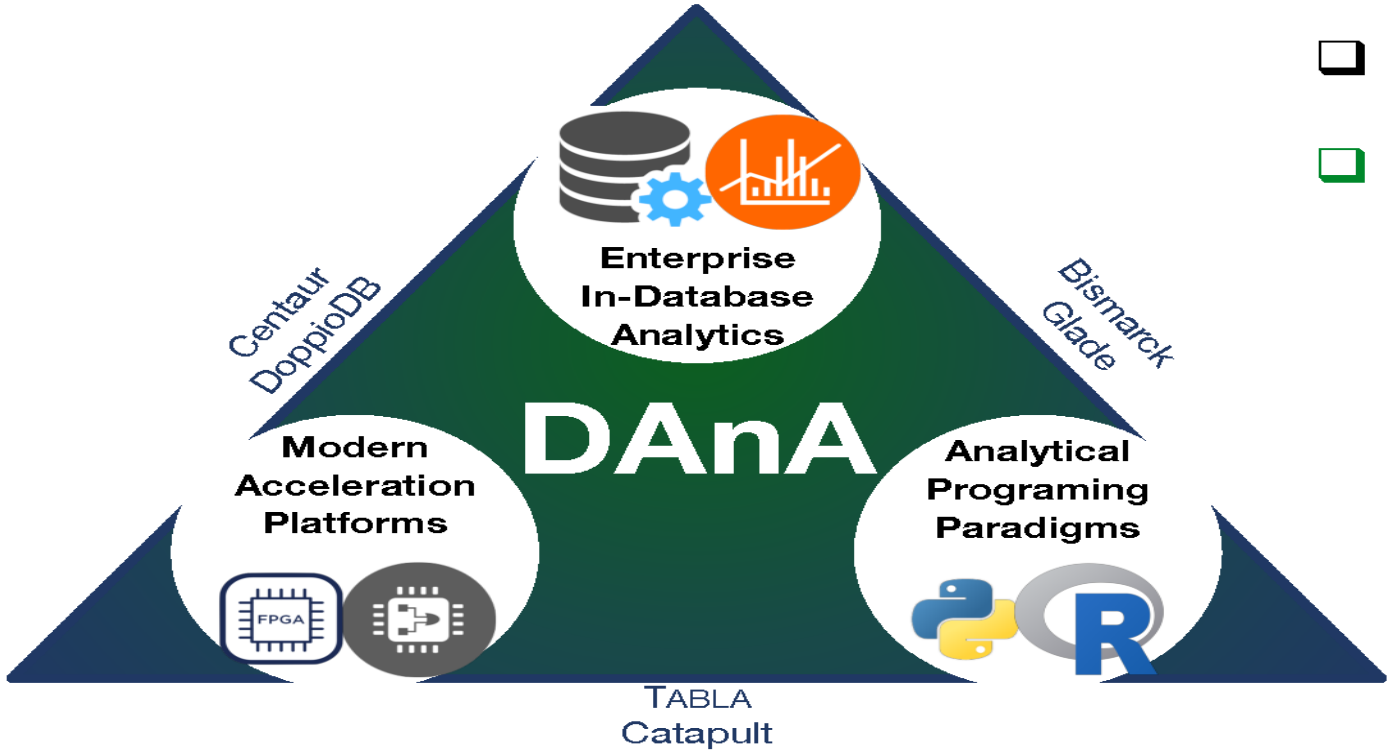


- Integration within Databases

In-RDBMS Advanced Analytics

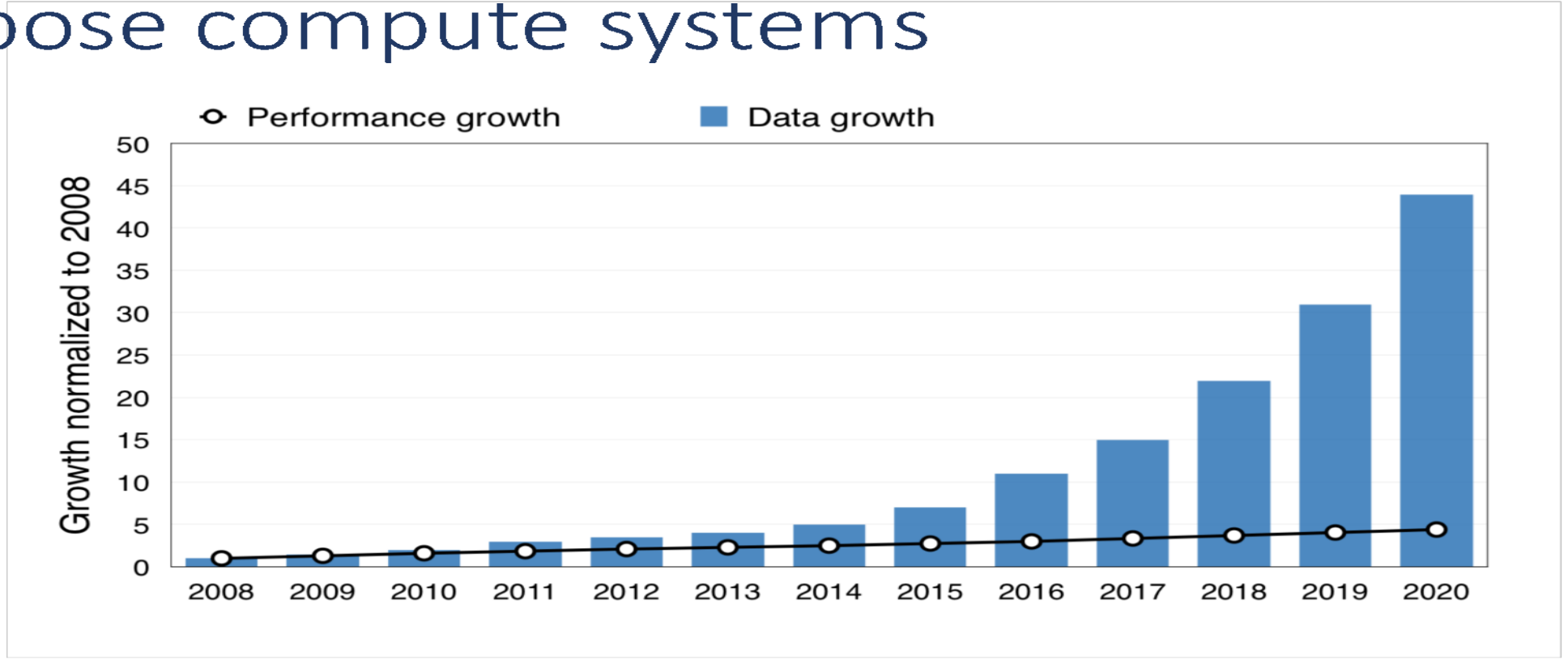


Objectives of **DAnA**



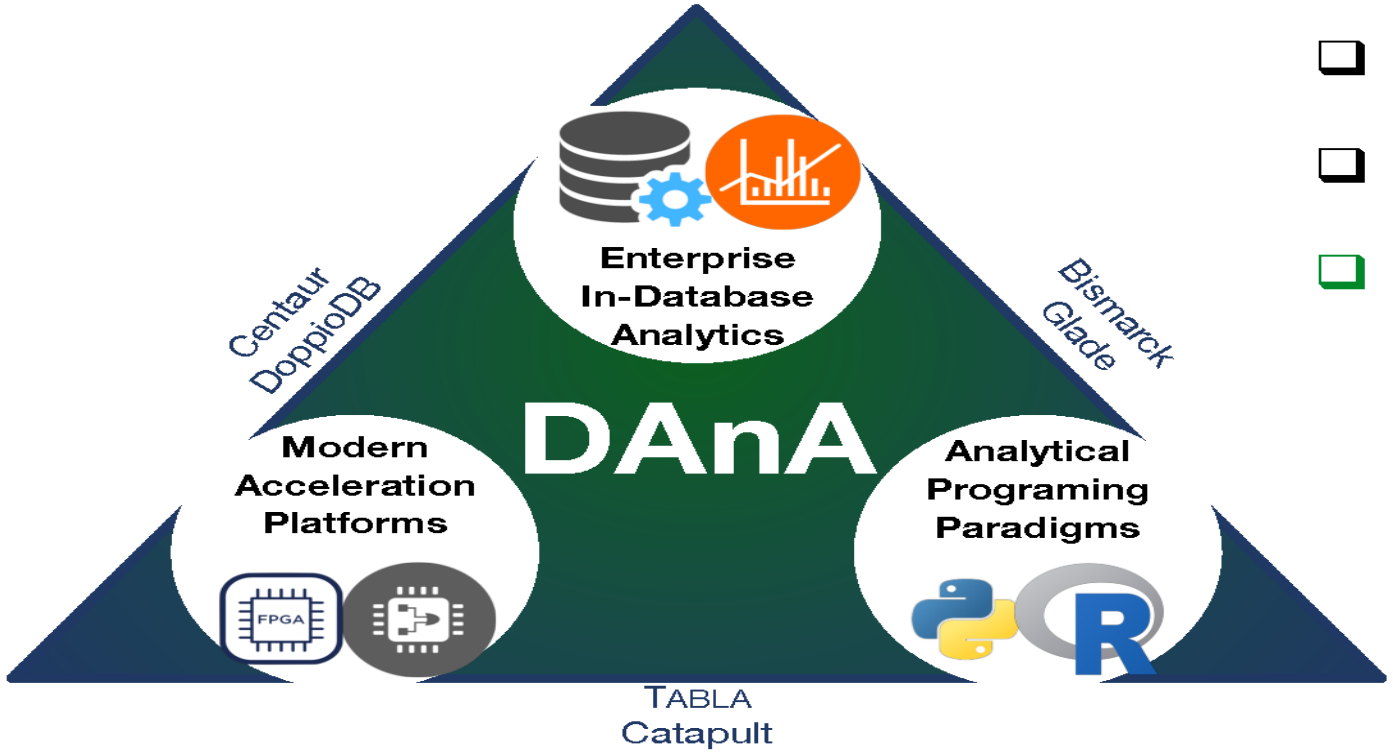
- Integration within Databases
- High-performance hardware acceleration

Data growth vs performance of general purpose compute systems



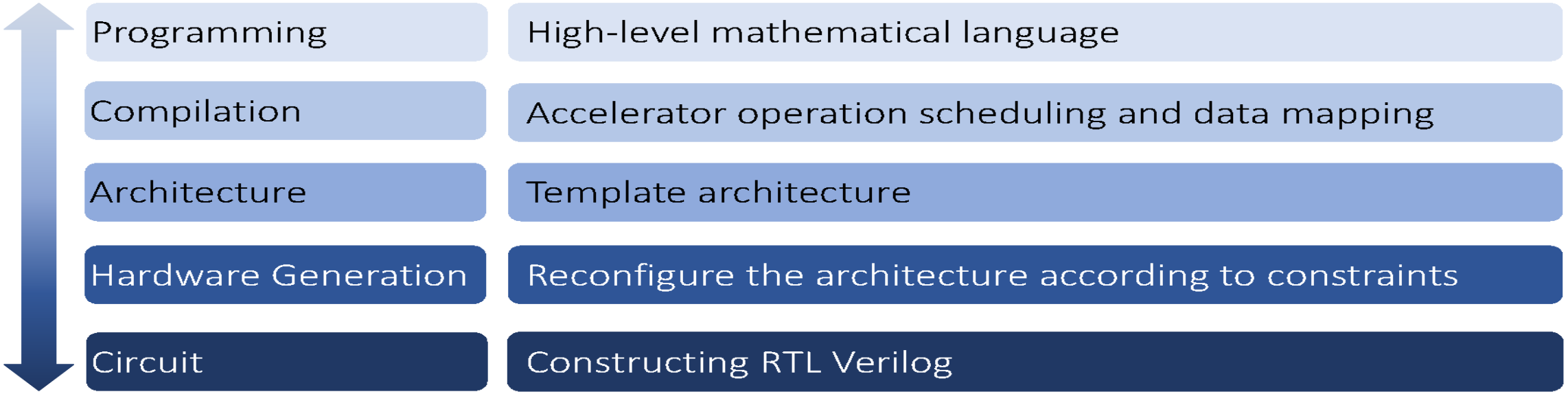
Source: Data growth trends: IDC's Digital Universe Study, December 2012
Performance growth trends: Esmaeilzadeh et al, "Dark Silicon and the End of Multicore Scaling," ISCA 2011

Objectives of **DAnA**



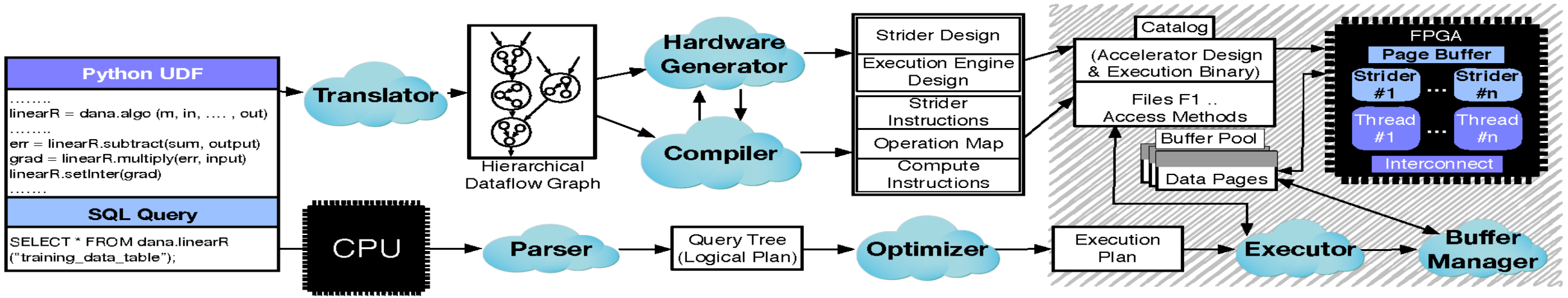
- Integration within Database
- High-performance hardware acceleration
- Expose a high-level programming interface

A Full-Stack Solution Towards Acceleration



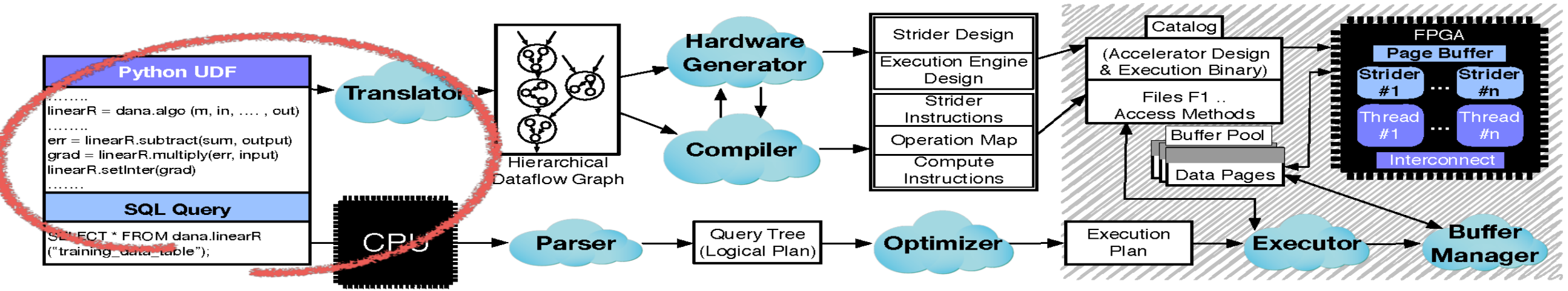
A Full-Stack Solution Towards Acceleration

Workflow



A Full-Stack Solution Towards Acceleration

Workflow



*http://www.postgresqltutorial.com/plpgsql-function-returns-a-table/

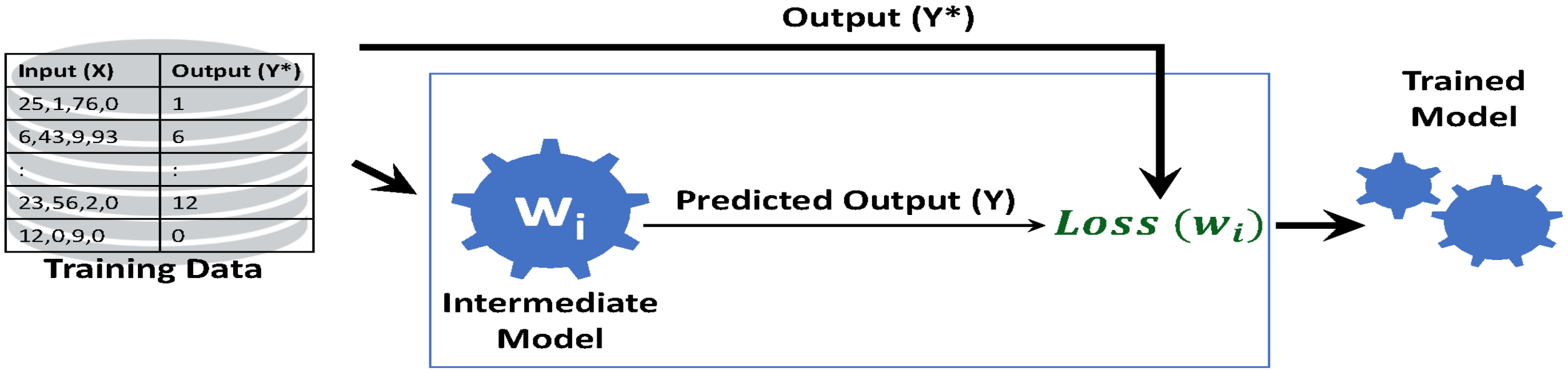
Using SQL to Specify the Training Data

```
SELECT * FROM dana.linearR('training_data_table');
```

A user defined function specifying a machine learning algorithm as an iterative update rule in our Python Embedded Domain Specific Language

Iterative Update Rules

Learning is solving an iterative optimization problem



$$find(w_i) \ni \{Loss(w_i) = \sum_i \|Y - Y^*\|\}\ is\ minimized$$

Programming Interface

Domain Specific Language in Python to specify the Machine Learning Algorithm

```
Update Rule
......
s = sigma (mo * in, 1)
er = s - out
grad = er * in

up = mu * grad
mo = mo - up

Merge Function
......
grad = merge(grad, mergeCoef, "+")

Convergence Criteria
setEpochs(10000)
```

User defined function

Table 1: Language constructs of DAnA's Python-embedded DSL.

Type	Keyword	Description
Component	algo	To specify an instance of the learning algorithm
Data Types	input	Algorithm input
	output	Algorithm output
	model	Machine learning model
	inter	Interim data type
	meta	Meta parameters
Mathematical Operations	+,-,*, /, >, <	Primary operations
	sigmoid, gaussian, sqrt	Non linear operations
	sigma, norm, pi	Group operations
Built-In Special Functions	merge(x, int, "operation")	Specify merge operation and number of merge instances
	setEpochs(int)	Set the maximum number of epochs
	setConvergence(x)	Specify the convergence criterion
	setModel(x)	Set the model variable

Programming Interface

Domain Specific Language in Python to specify the Machine Learning Algorithm

Update Rule

```
.....
s = sigma (mo * in, 1)
er = s - out
grad = er * in

up = mu * grad
mo = mo - up
```

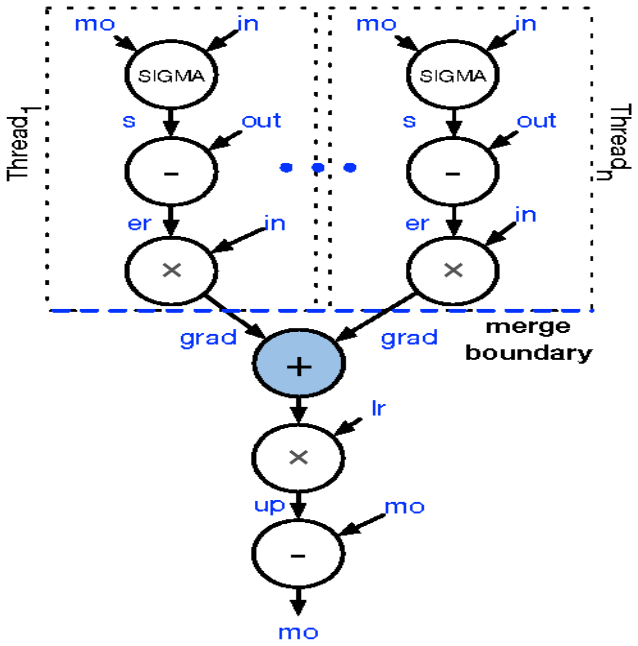
Merge Function

```
.....
grad = merge(grad, mergeCoef, "+")
```

Convergence Criteria

```
setEpochs(10000)
```

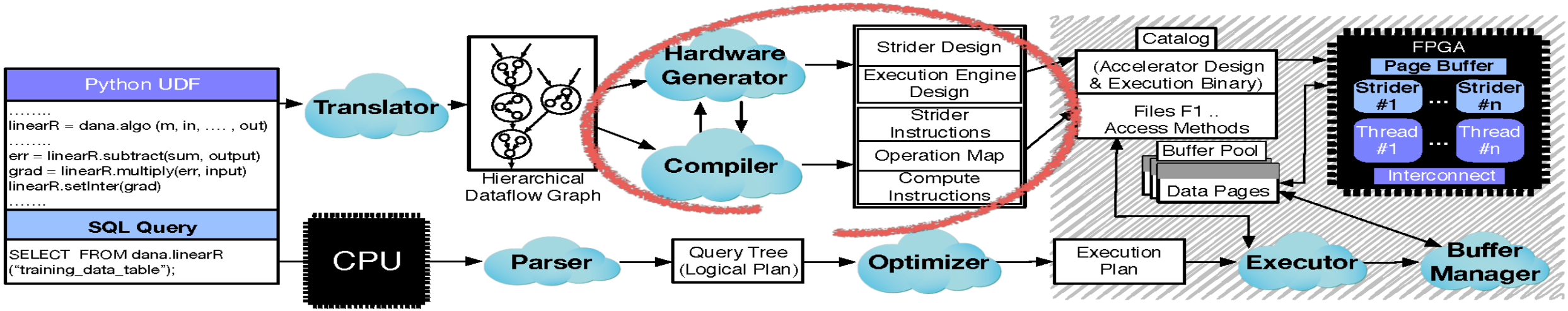
User defined function



Data Flow Graph

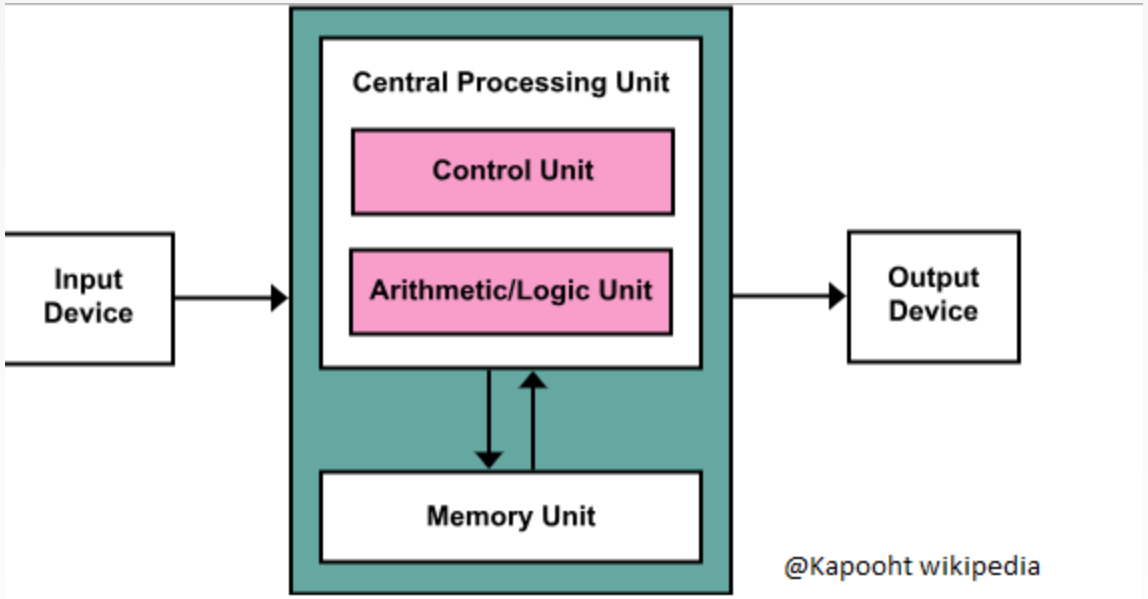
A Full-Stack Solution Towards Acceleration

Workflow



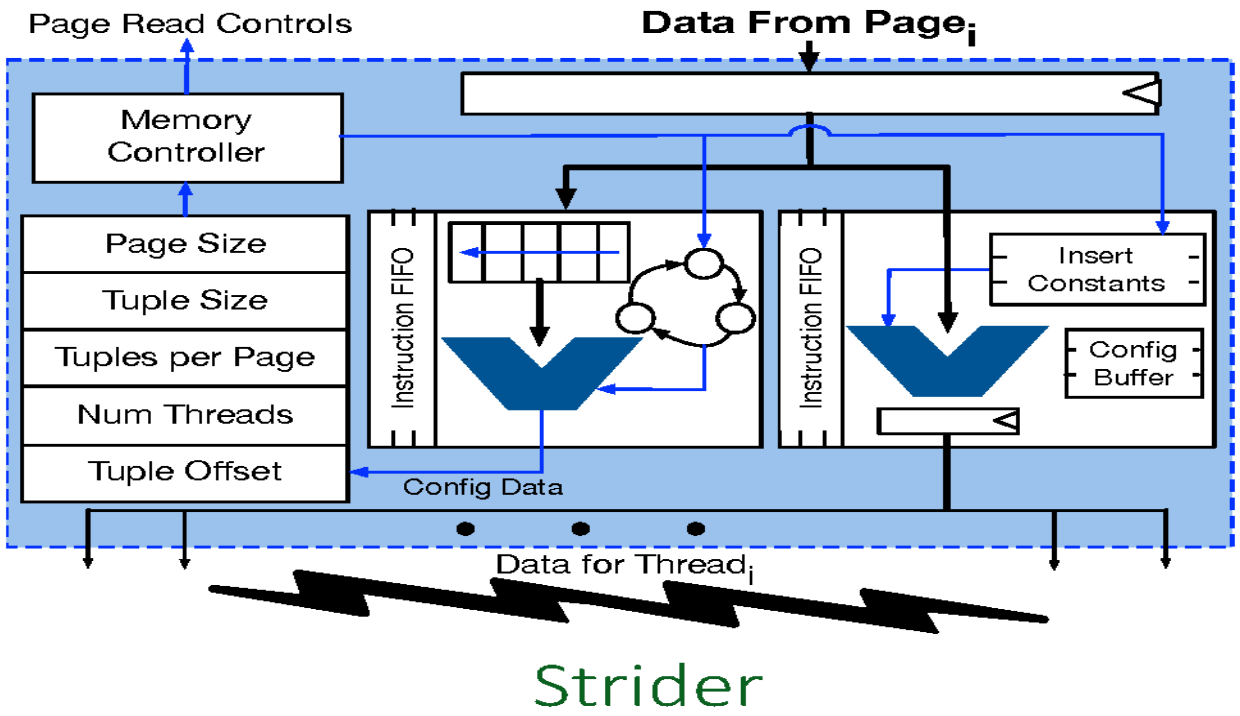
note

- Von Neumann architecture

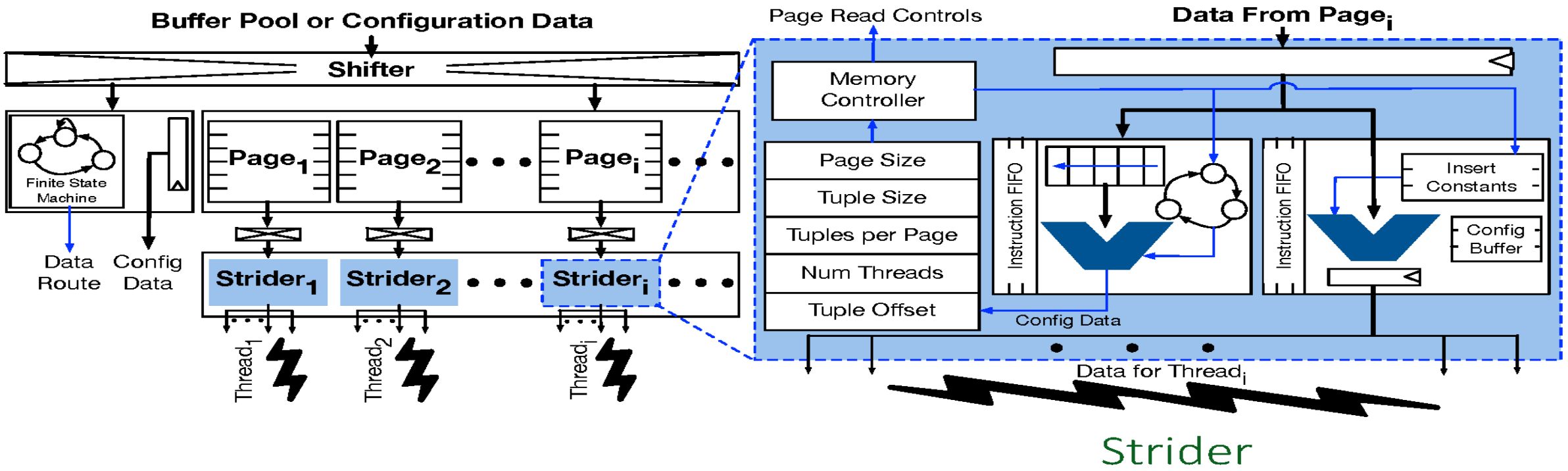


A Multi-threaded Data Access Engine

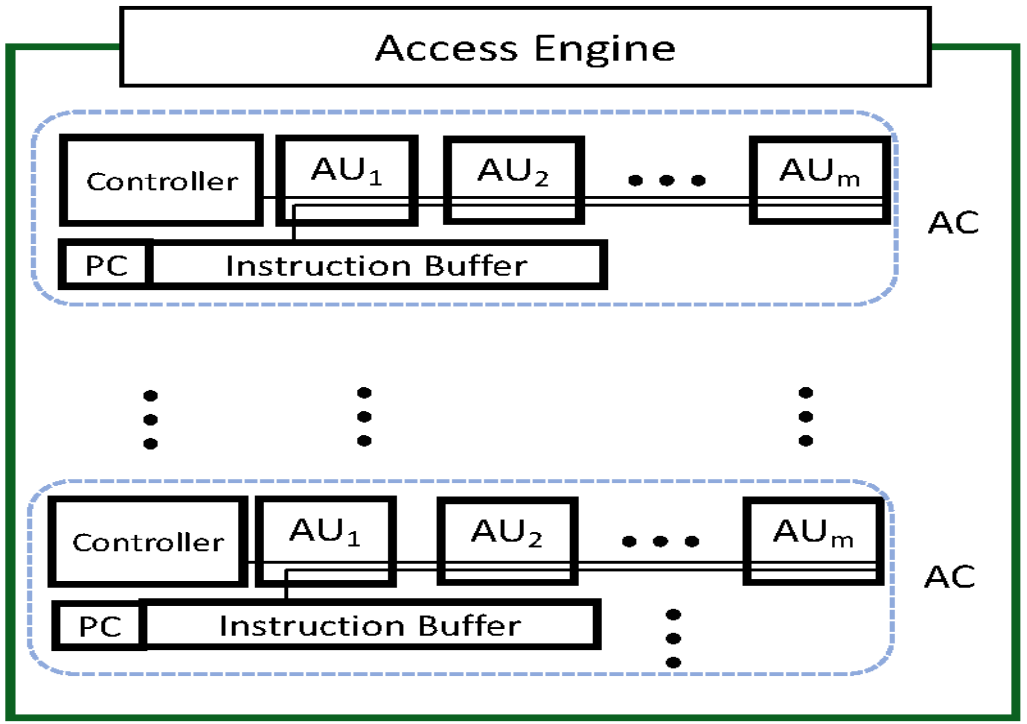
Striders are programmable using a domain specific instruction set architecture



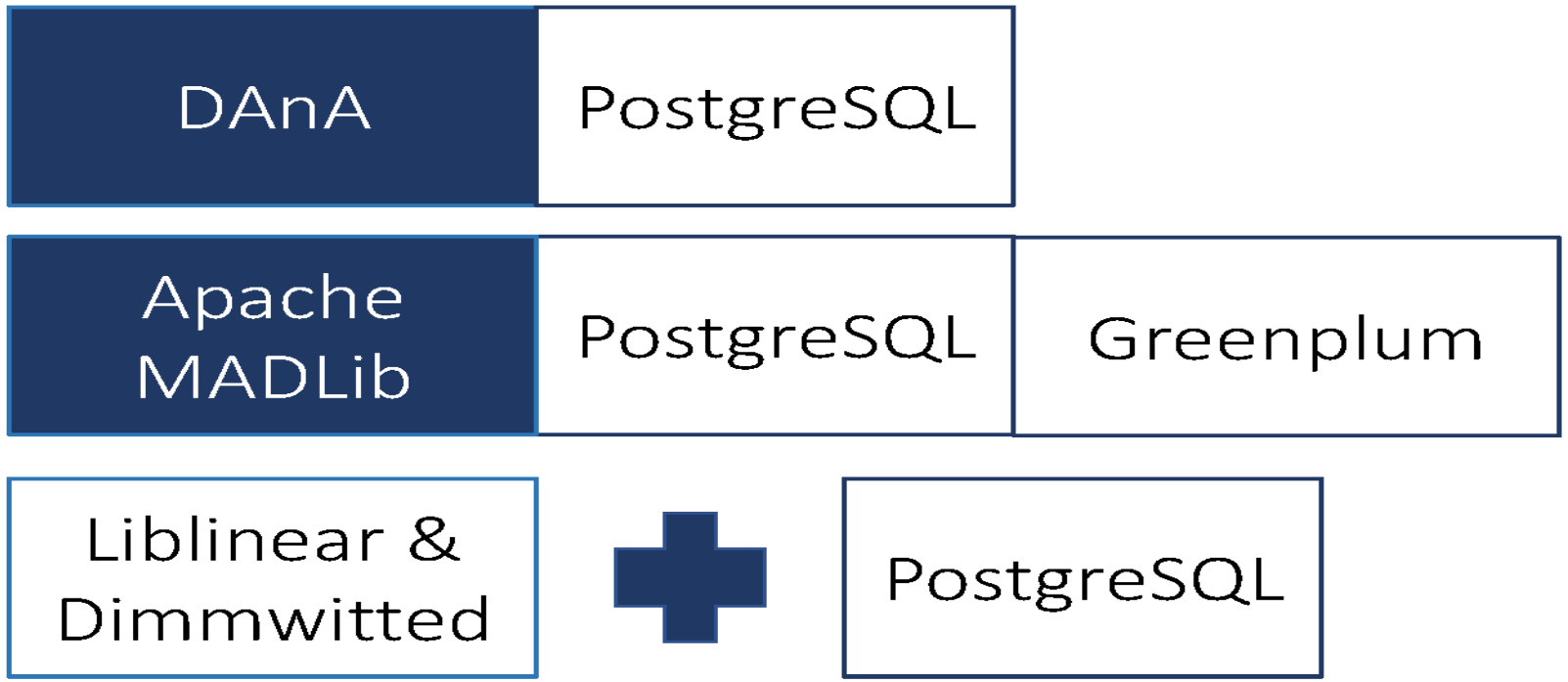
A Multi-threaded Data Access Engine



SIMD-MIMD High-Performance Dataflow Execution Engine



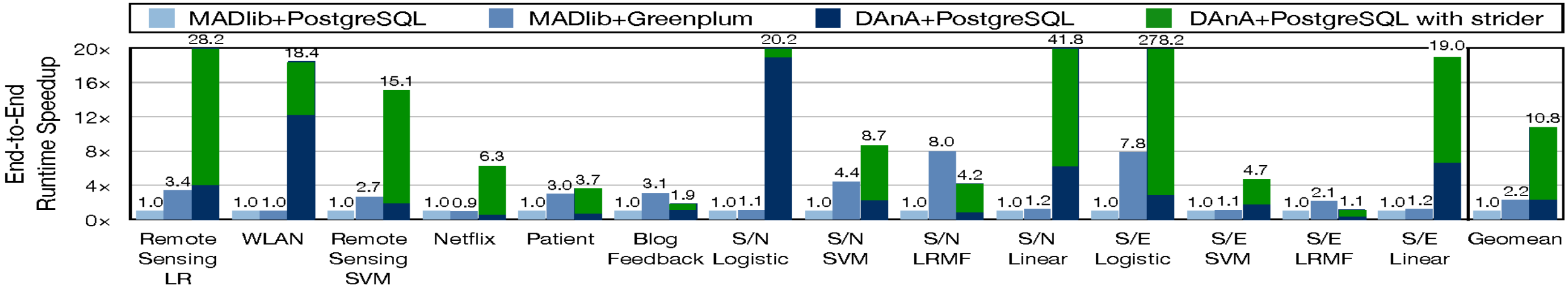
Integrate DAnA with PostgreSQL



Workload	Algorithm	Model Topology	# of Tuples	Training Data Size (MB)
Remote Sensing	Logistic Regression and Support Vector Machines	54	581,102	154
WLAN	Logistic Regression	520	19,937	42
Netflix	Low Rank Matrix Factorization	6,040; 3,952;10	6,040	96
Patient	Linear Regression	384	53,500	61
Blog Feedback	Linear Regression	280	52,397	84
S\N Logistic	Logistic Regression	2,000	387,944	3,031
S\N SVM	Support Vector Machines	1,740	678,392	5,300
S\N LRMF	Low Rank Matrix Factorization	19,880; 19,880; 10	19,880	1,587
S\N Linear	Linear Regression	8,000	130,503	4,087
S\E Logistic	Logistic Regression	6,033	1,044,024	25,292
S\E SVM	Support Vector Machines	7,129	1,356,784	38,840
S\E LRMF	Low Rank Matrix Factorization	28,002; 45,064; 10	45,064	5,067
S\E Linear	Linear Regression	8,000	1,000,000	32,124

Results

Comparison with MADlib an in-database analytics library

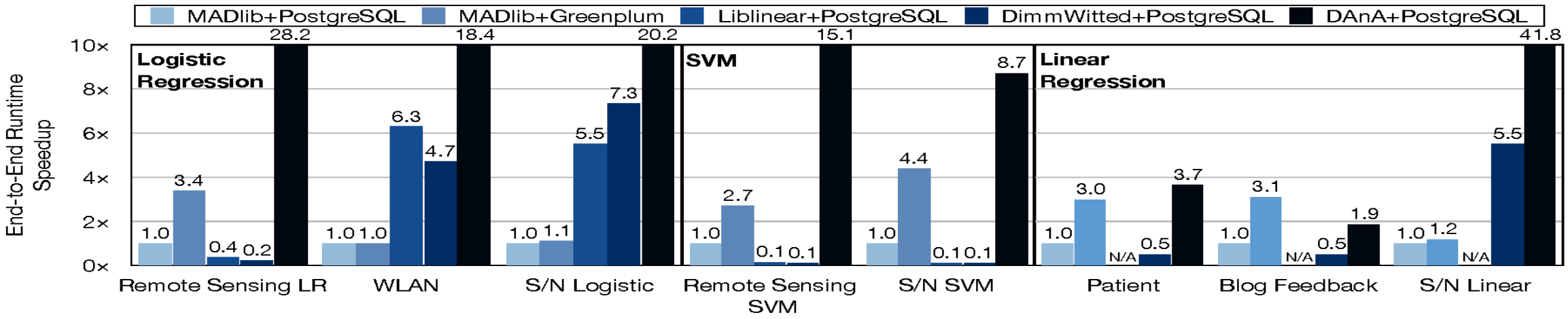


DAnA generated accelerators provide 10.8× Speedup / PostgreSQL+MADlib and 4.9× Speedup / Greenplum+MADlib

Striders contribute 4.7× speedup, as they directly integrate with the buffer pool of the RDBMS and bypass the CPU

Results

Comparison with Liblinear and Dimmwitted



DAnA provides 9.1× speedup over Liblinear and 10.4× over Dimmwitted

Strengths

- The authors recognized a connection between three seemingly unrelated fields of study and were able to bring them together to great effect.
- Domain specific language that bypasses Hardware description Languages (HDL).
- DNaN + Postgres has outperformed MADLib+Postgres and MADLib+GreenPlum, @ 8.3x. DNaN generated accelerators performed better than TABLA, an open source accelerator optimizer.
- The architecture of DAnA's execution engine allows DAnA to take advantage of data locality when it exists (e.g., when data must be transferred between different analytic units within a single analytic cluster), and spread out computation over many analytic units when data dependencies do not exist.

Weaknesses

- Is Domain Specific language and graph really needed for parallelization ? At the end they seems to depend on RDBMs pagination similarities for running instructions parallel on FPGA. Why not use existing MyHDL or other languages?
- RDBMS are generally used for OLTP database needs. In-RDBMS analytics may change RDBMS configuration space completely.
- There is no comparison with GPU (both cost and speed). GPUs are much cheaper than FPGA. $0.5 per core on a state of art GPU.
- Can Strider be used with Madlib? How will it perform?

Conclusion

DAnA

Exposes FPGA-based hardware acceleration to high-level database users

Next Step

Acceleration support for semi-structured data and column-based databases

Support compression and decompression in database pages

Extend the application base to support deep learning and unsupervised machine learning

Discussion