IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Analytics and Data Management: Fog Computing in IIoT

Dr. Sudip Misra

Professor

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Email: smisra@sit.iitkgp.ernet.in

Website: http://cse.iitkgp.ac.in/~smisra/

Research Lab: cse.iitkgp.ac.in/~smisra/swan/

Why Fog Computing for IIoT?

- Consistent release of data from sensors and machines
- Data may be critical as well as time-sensitive
- Need for immediate action and quick response
- Delay in proper action at proper time may create hazardous situation
- Major challenge is to handle the diversity: different protocols, different data syntax, different data source

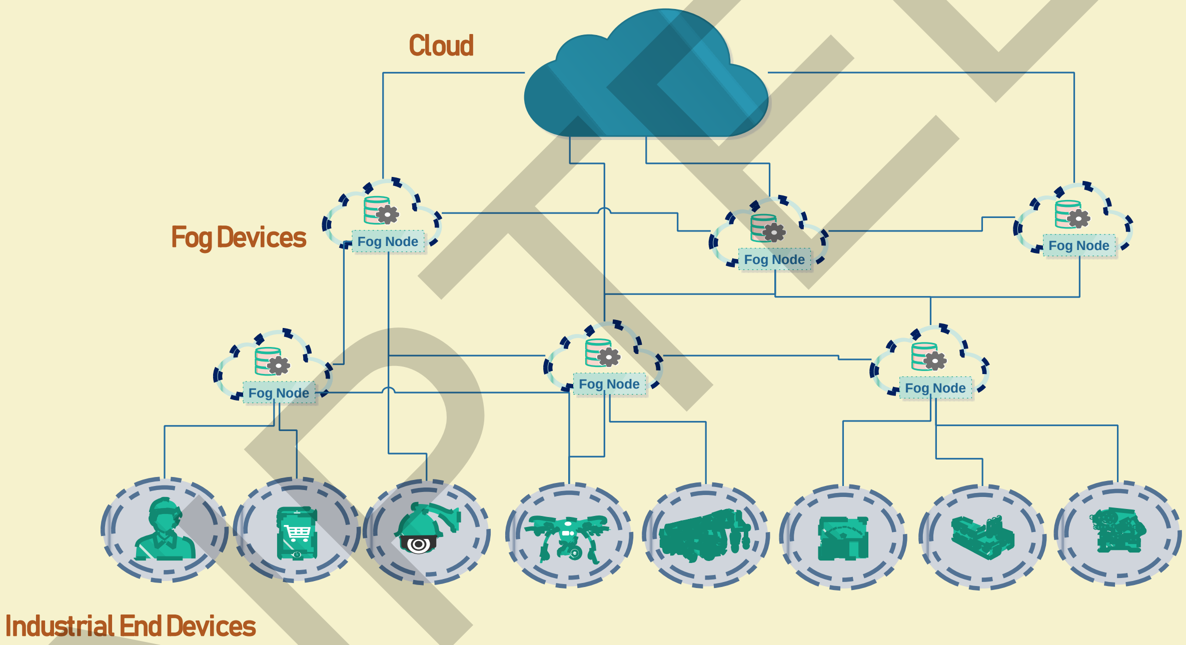
Source: Mohammad et al., 2018

Why Fog Computing for IIoT?

- Goal is to address the weaknesses of industrial automation
- Enabling new functionalities along with additional features
- Process control analytics
- Enriching the current functionalities

Source: Mohammad et al., 2018

Fog Computing Architecture for IIoT



Source: Mohammad et al., 2018

Fog Computing for Industrial Analytics

- Machine, process and data analysis in industries
- Advanced ways for optimized decision making and intelligent operations
- Achieving a new level of functioning and production in the system along with social values
- Classification can be done based on the place and function performed during analysis

Source: “Introduction to Edge Computing in IIoT”, Industrial Internet Consortium

Fog Computing for Industrial Analytics

- Support to algorithms at edge for real time control
- Additionally, high bandwidth communication and big data computations allows analysis on streaming data at cloud
- Prevention from unnecessary noisy big data crowd at cloud with prior filtration at edge

Source: "Introduction to Edge Computing in IIoT", Industrial Internet Consortium

Fog Enabled IIoT

- Real-time monitoring and visualization
- End to end security
- Scalable and flexible
- Reduced overall cost
- Novel trading ideas

Source: "Fog Computing pioneers", Nebbiolo Technologies

Use Cases

Use Cases

- Mining
- Smart Grid and Power
- Transportation
- Oil and Gas Industry

Source: Mohammad et al., 2018

Use Case - Mining

- Risky Environment
- IIoT may increase productivity and minimize over expenses
- Prediction and analysis of machines using IIoT reduces the operational cost
- Identifying the failure before it actually occurs
- Processing at fog nodes will increase accuracy

Source: Mohammad et al., 2018

Use Case - Smart Grid and Power Industry

- Dynamic demand of appliances
- Bi-directional communication between the consumer and supplier
- Power supply is provided from micro-grids, local distribution companies
- Advanced metering infrastructure for bi-directional communication

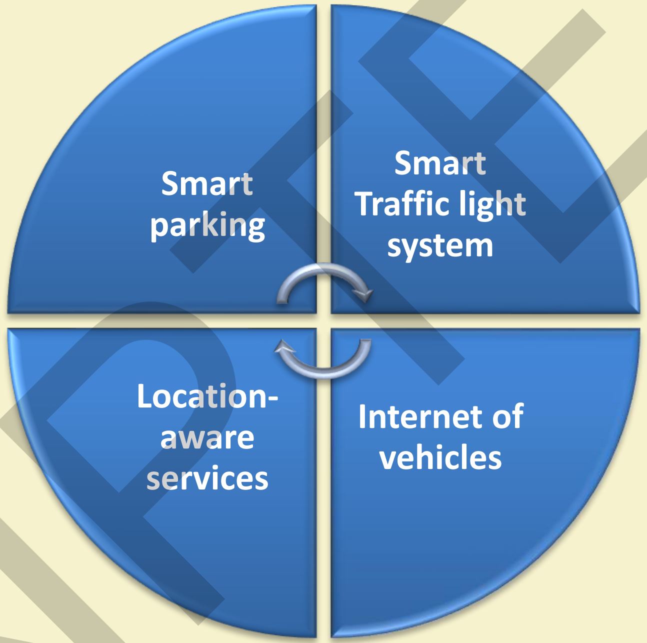
Source: Mohammad et al., 2018

Use Case - Smart Grid and Power Industry

- Continuous data exchange becomes a need
- Proper data communication is required
- Fog computing solves the issue

Source: Mohammad et al., 2018

Use Case - Transportation



Source: Mohammad et al., 2018

Use Case – Oil and Gas Industry

- Offering real-time advanced operation
- Detection of unusual events
- Step by step automation
- Real-time computation, control and management
- Support to scalability and adaptability

Source: "Fog Computing pioneers", Nebbiolo Technologies

IIoT Solutions using Fog

- Advanced hardware and software feature
  - Virtualization
  - Automation
  - Communication
  - Analysis
  - Prediction

Source: "Fog Computing pioneers", Nebbiolo Technologies

IIoT Solutions using Fog (Contd.)

- Asset management
  - Compliant cloud-fog analytics
  - Remotely managed machines
  - Energy management
  - Effective production
  - Quality with quantity

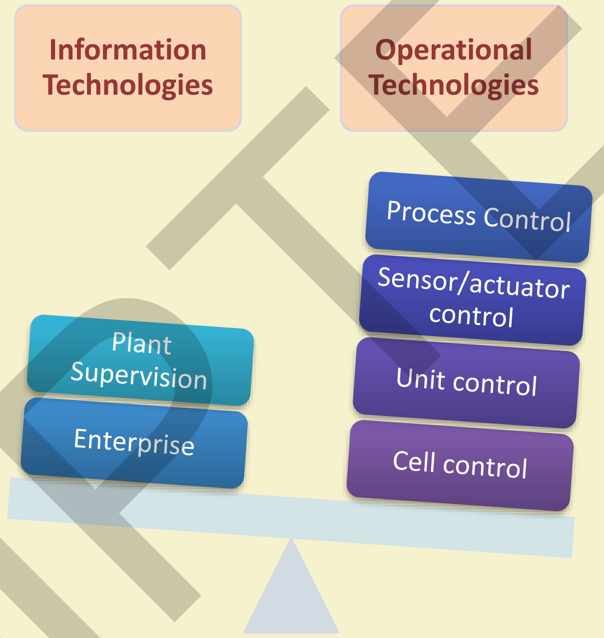
Source: “Fog Computing pioneers”, Nebbiolo Technologies

IIoT Solutions using Fog (Contd.)

- Futuristic monitoring and control system for industries
  - A platform for workload (real-time/non real-time) merging
  - Robust platform facilitating secure co-existence
  - Advanced fog-based control of IoT end points and sensors

Source: “Fog Computing pioneers”, Nebbiolo Technologies

Factors affecting Business



Source: “Fog Computing pioneers”, Nebbiolo Technologies

Fog platform Providers

- FogHorn
- Nebbiolo Technologies
- Crosser
- Sonm

FogHorn

- Edge network solution for quicker processing, analysing and responding
- Intelligent software platform for enabling edge computing
- Achieves efficient operation in lower cost
- Analysis and prediction on edge

Source: "Edge Intelligence software for IIoT", FogHorn Systems

Nebbiolo Technologies

- Bringing together IT and OT for real-time services
- Connecting the modern IT with future OT
- Optimized solution with smarter decision making capability in lower cost
- Products: fogOS, fogNode, fogSM

Source: "Fog Computing pioneers", Nebbiolo Technologies

Crosser

- Edge node software solution for asset data
- Supports any protocol, any PLC and any hardware
- Compute, Process and analyse wherever the requirement
- Real-time response to streaming IoT data
- Easy visual interfaces

Source: “Crosser Edge Computing Software”, Crosser

Sonm

- Distributed cloud services with fog as backend
- Either provide your hardware services or use third-party facility
- Current solutions
  - Blockchain infrastructure
  - Video streaming
  - Machine learning
  - Video rendering

Source: “SONM: Decentralized Fog Computing Platform”, Sonm

References

[1] A. Mohammad, S. Zeadally, and K. A. Harras, "Deploying Fog Computing in Industrial Internet of Things and Industry 4.0", *IEEE Transactions on Industrial Informatics*, 2018. **DOI:** 10.1109/TII.2018.2855198.

[2] S.Sarkar, S.Chaterjee and S.Misra, "Assessment of the Suitability of Fog Computing in the Context of Internet of Things", *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46-59, 2018.

[3] Bonomi, F., Milito, R.A., Natarajan, P., & Zhu, J. (2014). "Fog Computing: A Platform for Internet of Things and Analytics". *In Book*: *Big Data and Internet of Things*, pp. 169-186, Springer, 2014.

[4] A.V. Dastjerdi and R. Buyya, "Fog Computing: Helping the Internet of Things Realize Its Potential", *Computer*, 2016.

[5] "Fog Computing pioneers", Nebbiolo Technologies, Available Online: www.nebbiolo.tech, Accessed on August 16, 2018.

References

[6] "FogHorn Systems: Edge Intelligence software for IIoT", FogHorn Systems:, Available Online: www.foghorn.io, Accessed on August 16, 2018.

[7] "Crosser Edge Computing Software", Crosser, Available Online: crosser.io, Accessed on August 16, 2018.

[8] "SONM: Decentralized Fog Computing Platform", Sonm, Available Online: sonm.com, Accessed on August 16, 2018.

[9] "Introduction to Edge Computing in IIoT", Industrial Internet Consortium, Available Online: www.iiconsortium.org/pdf/Introduction_to_Edge_Computing_in_IIoT_2018-06-18.pdf, Accessed on August 23, 2018.

Thank You!!

IIT KHARAGPUR

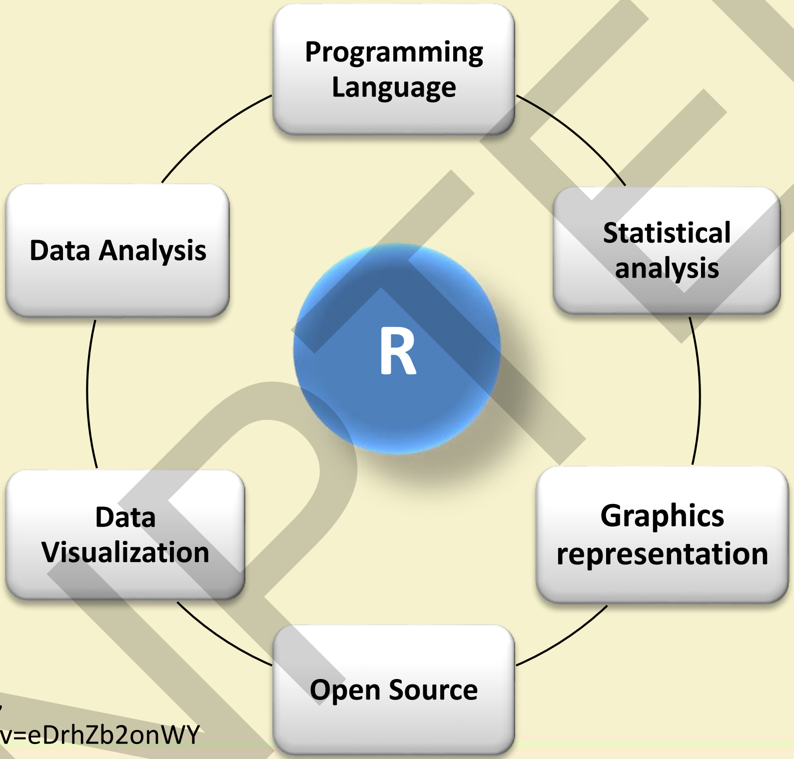
NPTEL ONLINE CERTIFICATION COURSES

IIoT Analytics and Data Management: Tutorial for R & Julia Programming

Dr. Sudip Misra
Professor
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Email: smisra@sit.iitkgp.ernet.in
Website: http://cse.iitkgp.ac.in/~smisra/
Research Lab: cse.iitkgp.ac.in/~smisra/swan/

R Programming

What is R tool ?



Source: R tutorial for beginners, Edureka,
URL: https://www.youtube.com/watch?v=eDrhZb2onWY

Fundamental concept of R

- Reserved words in R
- Variables in R
- R Operators
- R Data Types

Here all codes are run in RStudio Version 1.1.456 – © 2009-2018 RStudio, Inc in Windows 10

Reserved words in R

Words having special meaning and cannot be used as variable name, function name etc.

> ?reserved
or,
> help(reserved)

Reserved {base} R Documentation

Reserved Words in R

Description

The reserved words in R's parser are

`if` `else` `repeat` `while` `function` `for` `in` `next` `break`

`TRUE` `FALSE` `NULL` `Inf` `NaN` `NA` `NA_integer_` `NA_real_` `NA_complex_` `NA_character_`

`...` and `..1`, `..2` etc, which are used to refer to arguments passed down from a calling function. See the Introduction to R manual for usage of these syntactic elements, and dotsMethods for their use in formal methods.

Details

Reserved words outside quotes are always parsed to be references to the objects linked to in the 'Description', and hence they are not allowed as syntactic names (see `make.names`). They **are** allowed as non-syntactic names, e.g. inside backtick quotes.

[Package *base* version 3.4.1 Index]

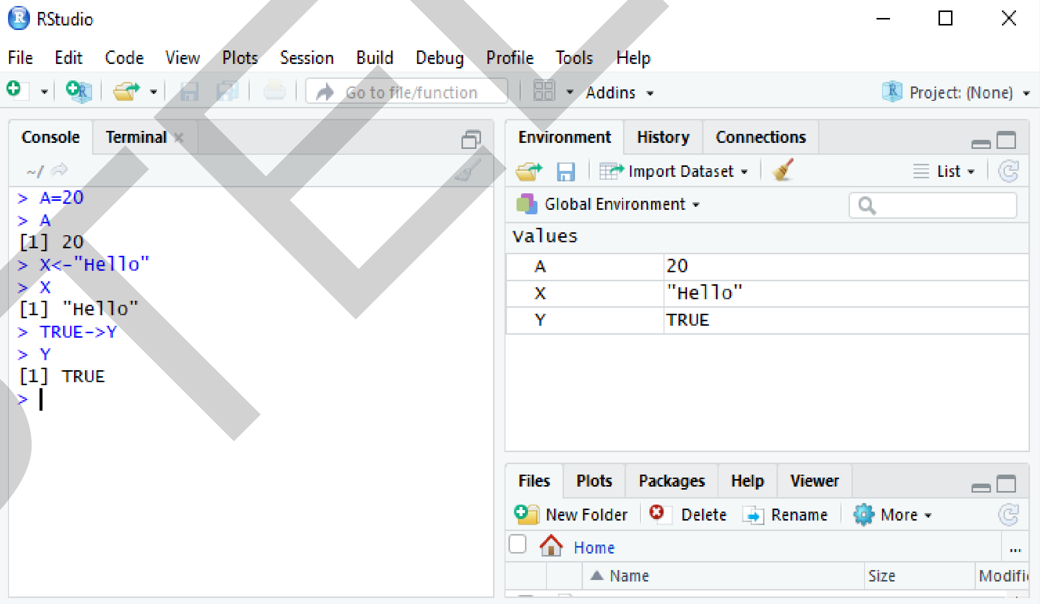
Source: R tutorial for beginners, Edureka,
URL: https://www.youtube.com/watch?v=eDrhZb2onWY

Figure is taken from RStudio Version 1.1.456 – © 2009-2018 RStudio, Inc.

Variables in R

- ➢ Declaration of variable do not need to specify the datatype
- ➢ Declaration of variables can be performed in three ways,

A=20;
X<-"Hello"
TRUE->Y



Source: R tutorial for beginners, Edureka,
URL: https://www.youtube.com/watch?v=eDrhZb2onWY

Figure is taken from RStudio Version 1.1.456 – © 2009-2018 RStudio, Inc.

R operators

1. Arithmetic Operators:

Arithmetic Operators	Purpose
+	Add two operators or unary plus
-	Subtract two operators or unary minus
*	Multiply two operators
/	Divide two operators
^	Left operand raised to the power of right
%%	Remainder of division
%/%	Divisions results in whole number adjusted to the left in the number line

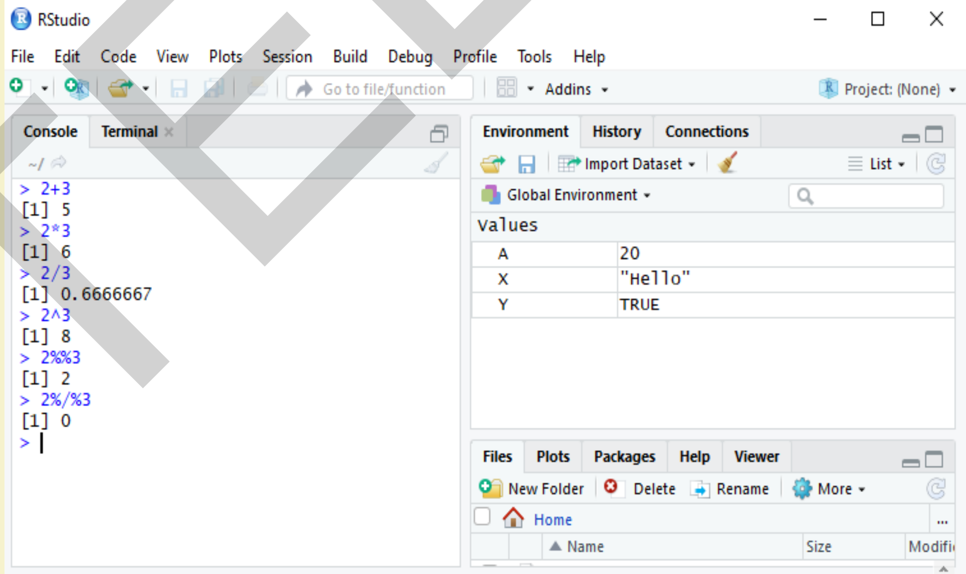


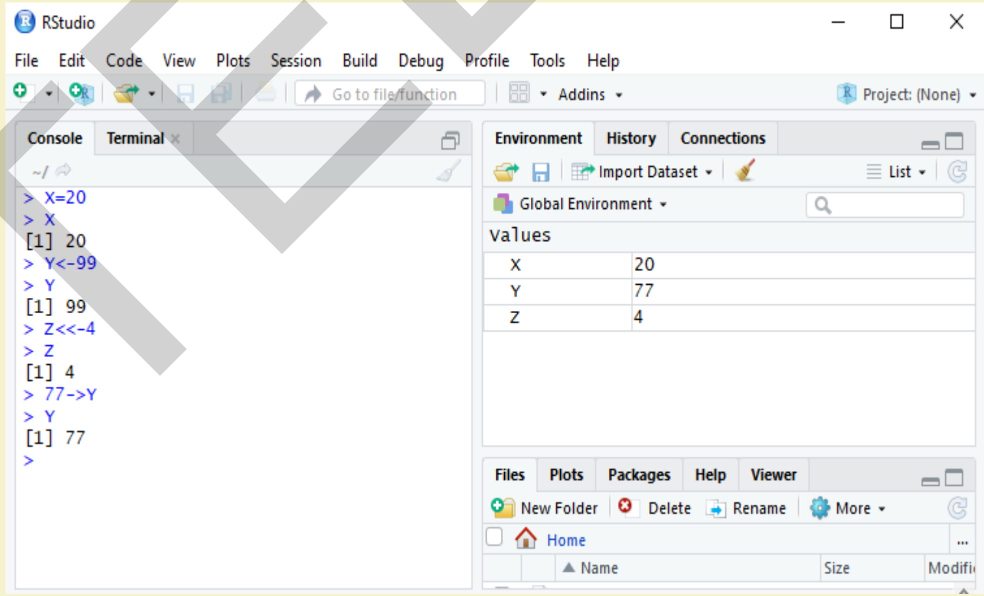
Figure is taken from RStudio Version 1.1.456 – © 2009-2018 RStudio, Inc.

Source: R tutorial for beginners, Edureka,
URL: https://www.youtube.com/watch?v=eDrhZb2onWY

R operators (Contd..)

2. Assignment Operators:

Assignment Operators	Purpose
=	variable= right operand
<-	variable<-right operand
<<-	variable<<-right operand
->	left operand->variable



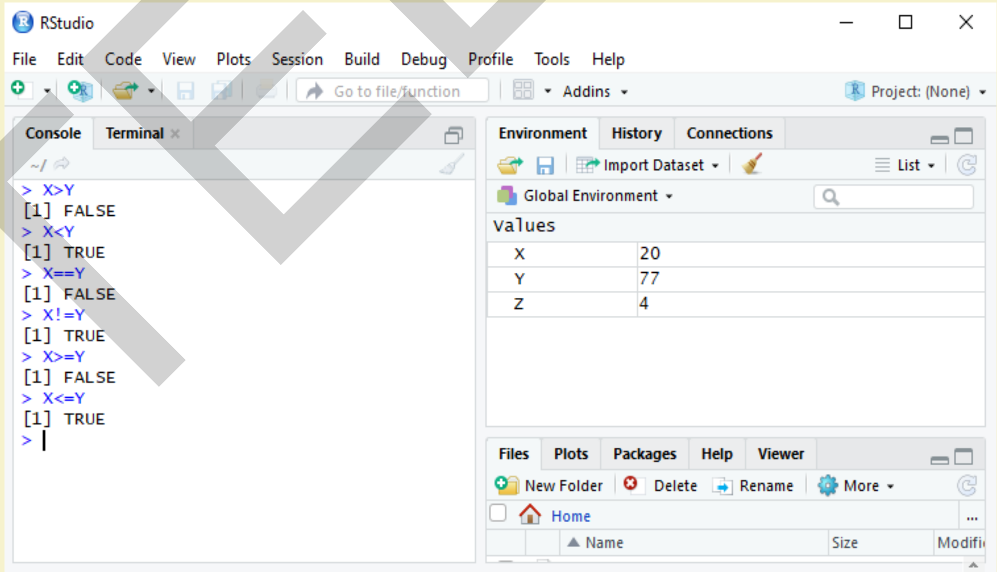
Source: R tutorial for beginners, Edureka,
URL: https://www.youtube.com/watch?v=eDrhZb2onWY

Figure is taken from RStudio Version 1.1.456 – © 2009-2018 RStudio, Inc.

R operators (Contd..)

3. Relational Operators:

Relational Operators	Purpose
>	Greater than operator
<	Less than operator
==	Equal to operator
!=	Not equal to operator
>=	Greater than and equal to
<=	Less than and equal to



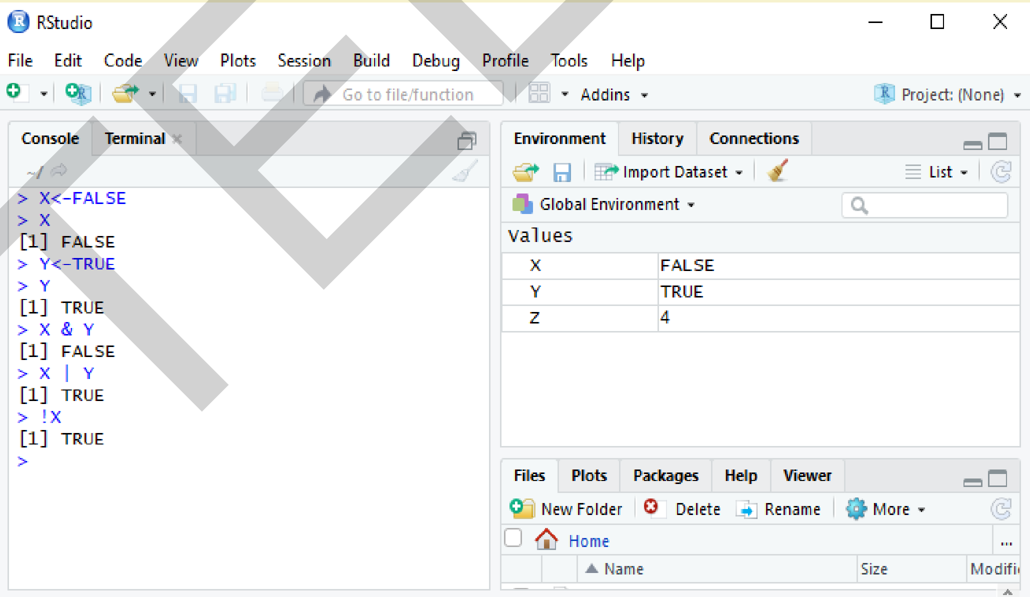
Source: R tutorial for beginners, Edureka,
URL: https://www.youtube.com/watch?v=eDrhZb2onWY

Figure is taken from RStudio Version 1.1.456 – © 2009-2018 RStudio, Inc.

R operators (Contd..)

4. Logical Operators:

Relational Operators	Purpose
&	AND operator
|	OR operator
!	NOT operator



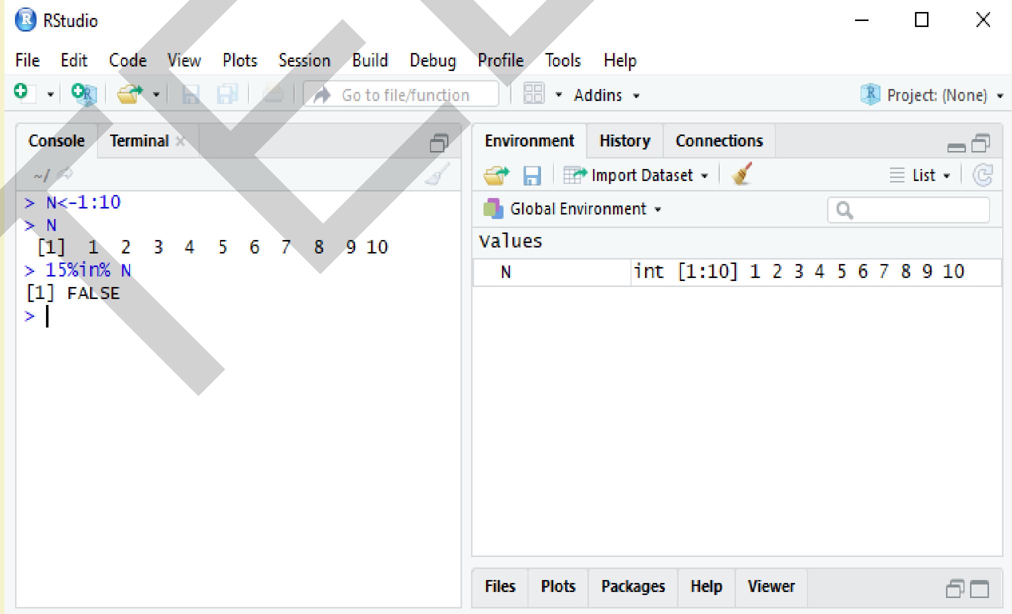
Source: R tutorial for beginners, Edureka,
URL: https://www.youtube.com/watch?v=eDrhZb2onWY

Figure is taken from RStudio Version 1.1.456 – © 2009-2018 RStudio, Inc.

R operators (Contd..)

5. Special Operators

Relational Operators	Purpose
:	Creates series of numbers for a vector
%in%	To check an element belongs to vector

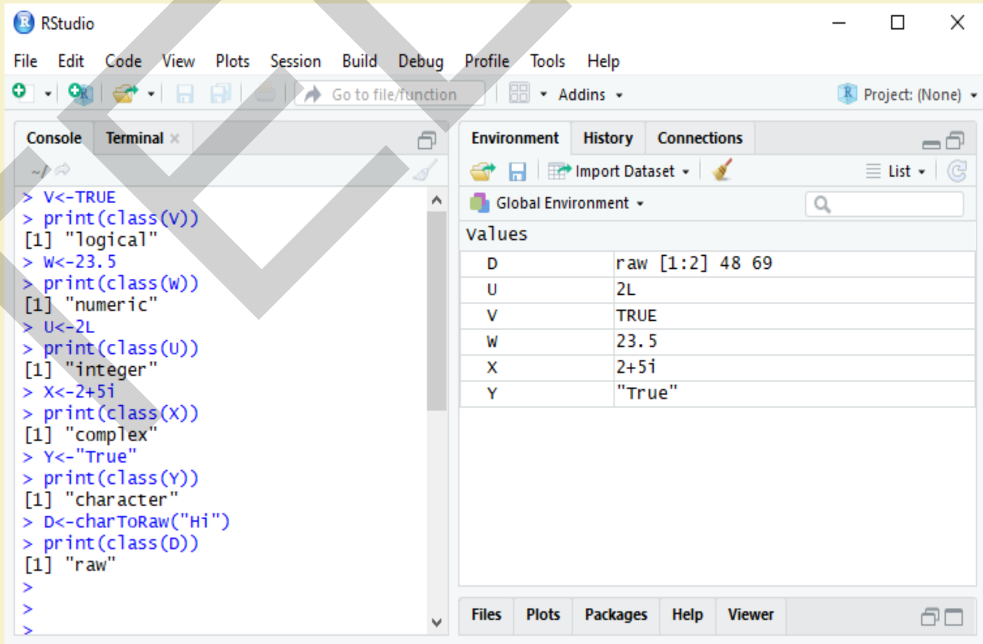


Source: R tutorial for beginners, Edureka,
URL: https://www.youtube.com/watch?v=eDrhZb2onWY

Figure is taken from RStudio Version 1.1.456 – © 2009-2018 RStudio, Inc.

R Data Types

- Unlike other programming languages like C and java, the variables are not declared as some data types in R
- Variables are assigned with R-objects and the data type of the R-object becomes the data type of variable
- Different objects are **Vectors, Lists, Matrices, Arrays, Factors, Data Frames**
- Different data types are **Logical, Numeric, Integer, Complex, Character, Raw**



Source: R tutorial for beginners, Edureka,
URL: https://www.youtube.com/watch?v=eDrhZb2onWY

Figure is taken from RStudio Version 1.1.456 – © 2009-2018 RStudio, Inc.

R Data Types (Contd..)

- Vectors

```
> apple<-c('red', 'green’, 'yellow’)
> print(apple)
[1] "red" "green" "yellow"
```

- Arrays

```
a<-array(c('green','yellow'),dim=c(3,3,2))
> print(a)
 , , 1

     [,1] [,2] [,3]
[1,] "green" "yellow" "green"
[2,] "yellow" "green" "yellow"
[3,] "green" "yellow" "green“

 , , 2
 [,1] [,2] [,3]
 [1,] "yellow" "green" "yellow“
 [2,] "green" "yellow" "green“
 [3,] "yellow" "green" "yellow"
```

- Matrices

```
> Mat=matrix(c('a','b','c','d','e','f'), nrow=2,
ncol=3, byrow=TRUE)
> print(Mat)
    [,1] [,2] [,3]
[1,] "a" "b" "c"
[2,] "d" "e" "f"
```

- Lists

```
> list1<-list(c(2,5,3),21.3,sin)
> print(list1)
[[1]] [1] 2 5 3
[[2]] [1] 21.3
[[3]] function (x) .Primitive("sin")
```

Source: R tutorial for beginners, Edureka,
URL: https://www.youtube.com/watch?v=eDrhZb2onWY

Figures are taken from Rstudio Version 1.1.456 – © 2009-2018 RStudio, Inc.

Important machine learning packages of R

Packages	Functions
1. **e1071**	Fuzzy clustering, support vector machine, naïve bayes classifier etc
1. **rpart**	Regression tree etc
1. **nnet**	Feed forward neural network etc
1. **randomForest**	Random forests for classification and regression
1. **igraph**	Network analysis tools
1. **caret**	Functions for creating predictive models

Execution of machine learning

- Install caret package using,

```
install.packages("caret", dependencies=c("Depends", "Suggests"))
```

or

```
install.packages("caret")
```

- 2. Load the packages using,

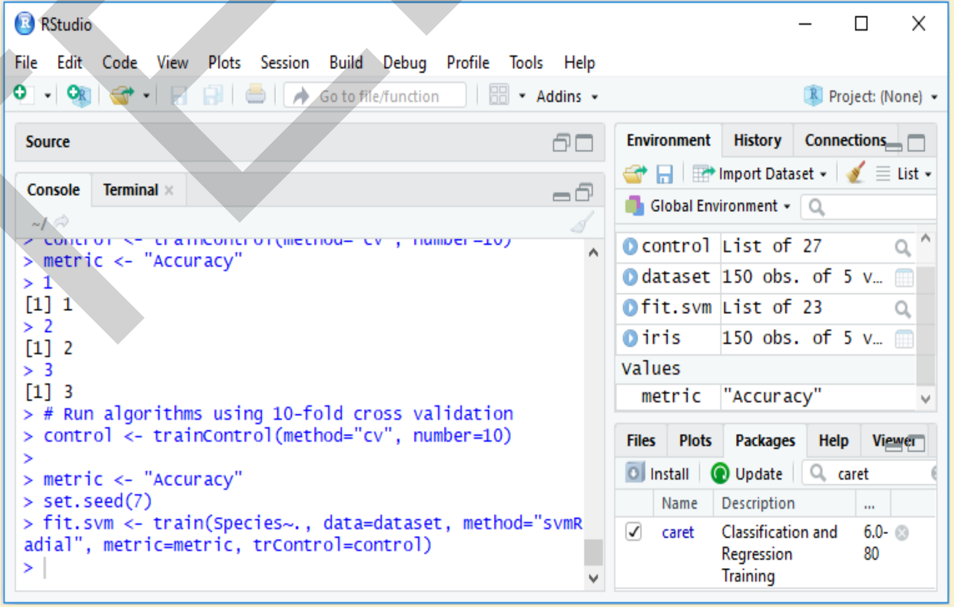
```
> library(caret)
```

- 3. Load data using

```
> data("iris")
```

- 4. Rename the dataset

```
> dataset <- iris
```



Source: Your First Machine Learning Project in R Step-By-Step
URL: https://machinelearningmastery.com/machine-learning-in-r-step-by-step/

Execution of machine learning (contd...)

- 10 fold cross validation to estimate accuracy

```
> # Run algorithms using 10-fold cross validation
> control <- trainControl(method="cv", number=10)
>
> metric <- "Accuracy"
```

- Support vector machine with linear kernel

```
> set.seed(7)
> fit.svm <- train(Species~., data=dataset, method="svmRadial", metric=metric, trControl=control)
```

Source: Your First Machine Learning Project in R Step-By-Step
URL: https://machinelearningmastery.com/machine-learning-in-r-step-by-step/

Julia Programming

Why Julia programming?

Julia merges python's benefits with c's performance

- Open source
- Distributed computation and parallelism possible
- Support efficiently Unicode
- Call c functions directly

Source: Julia tutorial URL:http://codebasicshub.com/
Source: Julia 1.0 Documentation URL: https://docs.julialang.org/en/stable/

Basics of Julia programming

➤ Use println() is used to print

```
In [1]: println("I'm excited to learn Julia!")
        I'm excited to learn Julia!
```

➤ Variables can be assigned without defining the type

```
In [2]: my_answer = 42
        typeof(my_answer)
Out[2]: Int64

In [4]: my_pi = 3.14159
        typeof(my_pi)
Out[4]: Float64
```

➤ Basic math

```
In [5]: sum = 3 + 7
Out[5]: 10

In [6]: difference = 10 - 3
Out[6]: 7

In [7]: product = 20 * 5
Out[7]: 100

In [8]: quotient = 100 / 10
Out[8]: 10.0

In [9]: power = 10 ^ 2
Out[9]: 100

In [10]: modulus = 101 % 2
Out[10]: 1
```

codes are run in browser on JuliaBox.com
https://www.juliabox.com/notebook/notebooks/tutorials/intro-to-julia/03.%20Data%20structures.ipynb

Basics of Julia programming (Contd...)

- Assigning string

```
In [1]: s1 = "I am a string."
Out[1]: "I am a string."
```

- Use of $ sign for string interpolation

```
In [3]: name = "Jane"
        num_fingers = 10
        num_toes = 10
Out[3]: 10

In [4]: println("Hello, my name is $name.")
        println("I have $num_fingers fingers and $num_toes toes.")

        Hello, my name is Jane.
        I have 10 fingers and 10 toes.
```

- String concatenation

```
In [5]: s3 = "How many cats ";
        s4 = "is too many cats?";

In [6]: string(s3, s4)
Out[6]: "How many cats is too many cats?"
```

codes are run in browser on JuliaBox.com
https://www.juliabox.com/notebook/notebooks/tutorials/intro-to-julia/03.%20Data%20structures.ipynb

Basics of Julia programming (Contd...)

➢ Data structures

1. Tuples

```
In [1]: myfavoriteanimals = ("penguins", "cats", "sugargliders")
Out[1]: ("penguins", "cats", "sugargliders")
```

We can index into this tuple,

```
In [2]: myfavoriteanimals[1]
Out[2]: "penguins"
```

but since tuples are immutable, we can't update it

```
In [3]: myfavoriteanimals[1] = "otters"
MethodError: no method matching setindex!(::Tuple{String,String,String}, ::String, ::Int64)
```

codes are run in browser on JuliaBox.com
https://www.juliabox.com/notebook/notebooks/tutorials/intro-to-julia/03.%20Data%20structures.ipynb

Basics of Julia programming (Contd...)

2. Dictionary

- Dict() is used for creating dictionaries

```
In [4]:  myphonebook = Dict("Jenny" => "867-5309", "Ghostbusters" => "555-2368")

Out[4]:  Dict{String,String} with 2 entries:
           "Jenny"        => "867-5309"
           "Ghostbusters" => "555-2368"
```

- Show a particular instance

```
In [5]:  myphonebook["Jenny"]

Out[5]:  "867-5309"
```

codes are run in browser on JuliaBox.com
https://www.juliabox.com/notebook/notebooks/tutorials/intro-to-julia/03.%20Data%20structures.ipynb

Basics of Julia programming (Contd...)

3. Arrays

```
In [6]: myfriends = ["Ted", "Robyn", "Barney", "Lily", "Marshall"]

Out[6]: 5-element Array{String,1}:
         "Ted"
         "Robyn"
         "Barney"
         "Lily"
         "Marshall"
```

```
In [7]: fibonacci = [1, 1, 2, 3, 5, 8, 13]

Out[7]: 7-element Array{Int64,1}:
          1
          1
          2
          3
          5
          8
         13
```

codes are run in browser on JuliaBox.com
https://www.juliabox.com/notebook/notebooks/tutorials/intro-to-julia/03.%20Data%20structures.ipynb

References

[1] R tutorial for beginners, Edureka,
URL: https://www.edureka.co/blog/r-tutorial/?utm_source=youtube&utm_campaign=r-tutorial-020617-wr&utm_medium=description

[2] Your First Machine Learning Project in R Step-By-Step
URL: https://machinelearningmastery.com/machine-learning-in-r-step-by-step/

[2] Julia tutorial URL:http://codebasicshub.com/

[3] Julia 1.0 Documentation URL: https://docs.julialang.org/en/stable/

[4] : codes are run in browser on JuliaBox.com
https://www.juliabox.com/notebook/notebooks/tutorials/intro-to-julia/03.%20Data%20structures.ipynb

Thank You!!

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

NPTEL

IIoT Analytics and Data Management: Data Management with Hadoop

Dr. Sudip Misra

Professor

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Email: smisra@sit.iitkgp.ernet.in

Website: http://cse.iitkgp.ac.in/~smisra/

Research Lab: cse.iitkgp.ac.in/~smisra/swan/

What is Data Management

- Data Management
  - Ensures that research data is stored, archived or disposed off in a safe and secure manner during and after the conclusion of a research project
  - Includes the development of policies and procedures to manage data handled electronically as well as through non-electronic means

- In recent days, most industrial data –
  - Big Data
    - Due to heavy traffic generated by IoT devices
    - Huge amount of data generated by the deployed sensors

Data Management: Technologies

- Cloud computing
  - Essential characteristics according to NIST
    - On-demand self service
    - Broad network access
    - Resource pooling
    - Rapid elasticity
    - Measured service
  - Basic service models provided by cloud computing
    - Infrastructure-as-a-Service (IaaS)
    - Platform-as-a-Service (PaaS)
    - Software-as-a-Service (SaaS)

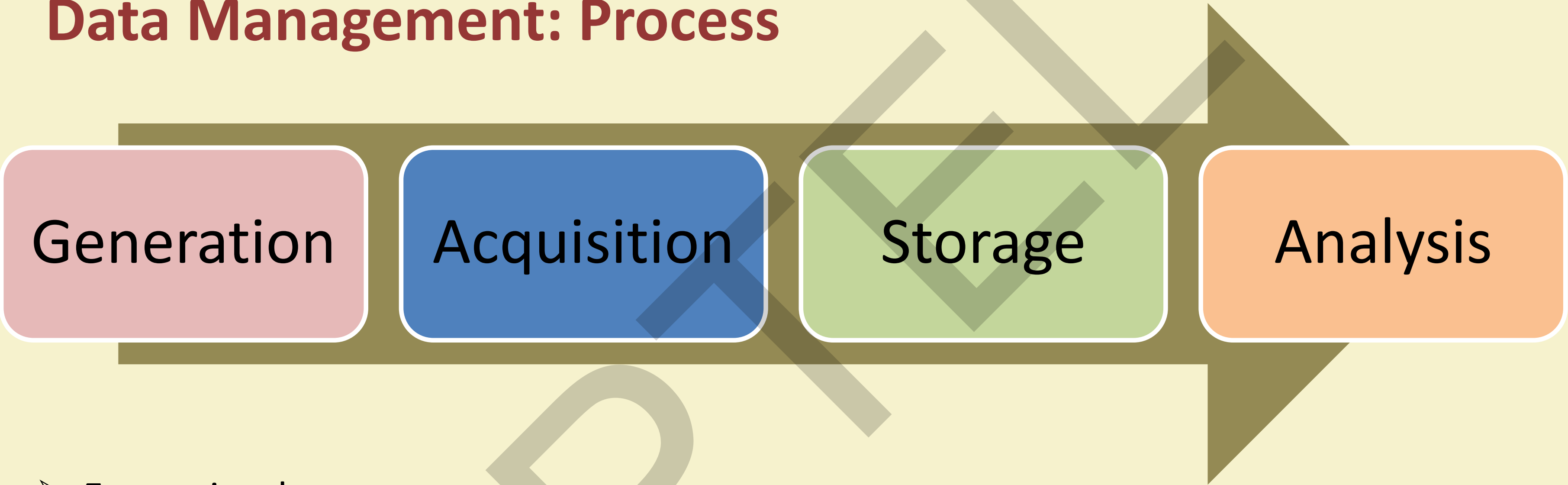
Data Management: Technologies (Contd.)

- Internet of Things (IoT) and Big Data
  - According to Techopedia, IoT *"describes a future where every day physical objects will be connected to the internet and will be able to identify themselves to other devices."*
  - Sensors embedded into various devices and machines and deployed into fields.
  - Sensors transmit sensed data to remote servers via Internet.
  - Continuous data acquisition from mobile equipment, transportation facilities, public facilities, and home appliances

Data Management: Technologies (Contd.)

- Data handling at data centers
  - Storing, managing, and organizing data.
  - Estimates and provides necessary processing capacity.
  - Provides sufficient network infrastructure.
  - Effectively manages energy consumption.
  - Replicates data to keep backup.
  - Develop business oriented strategic solutions from big data.
  - Helps business personnel to analyze existing data.
  - Discovers problems in business operations.

Data Management: Process



- Enterprise data
- IoT data
- Bio-medical data
- Other data

- Data collection
- Data transportation
- Data pre-processing

- Hadoop
- MapReduce
- NoSQL databases

- Bloom filter
- Parallel computing
- Hashing and indexing

Data Sources

- Enterprise data
  - Online trading and analysis data.
  - Production and inventory data.
  - Sales and other financial data.
- IoT data
  - Data from industry, agriculture, traffic, transportation
  - Medical- care data,
  - Data from public departments, and families.
- Bio-medical data
  - Masses of data generated by gene sequencing.
  - Data from medical clinics and medical R&Ds.
- Other fields
  - Fields such as – computational biology, astronomy, nuclear research etc

Data Acquisition

- Data collection
  - Log files or record files that are automatically generated by data sources to record activities for further analysis.
  - Sensory data such as sound wave, voice, vibration, automobile, chemical, current, weather, pressure, temperature etc.
  - Complex and variety of data collection through mobile devices. E.g. – geographical location, 2D barcodes, pictures, videos etc.

Data Acquisition

- Data transmission
  - After collecting data, it will be transferred to storage system for further processing and analysis of the data.
  - Data transmission can be categorized as – Inter-DCN transmission and Intra-DCN transmission.

Data Acquisition (Contd.)

- Data pre-processing
  - Collected datasets suffer from noise, redundancy, inconsistency etc.
  - Pre-processing of relational data mainly follows – integration, cleaning, and redundancy mitigation
  - Integration is combining data from various sources and provides users with a uniform view of data.
  - Cleaning is identifying inaccurate, incomplete, or unreasonable data, and then modifying or deleting such data.
  - Redundancy mitigation is eliminating data repetition through detection, filtering and compression of data to avoid unnecessary transmission.

Data Storage

- Databases
  - Emergence of non-traditional relational databases (NoSQL) in order to deal with the characteristics that big data possess.
  - Three main NoSQL databases – Key-value databases, column-oriented databases, and document-oriented databases.

Data Storage (Contd.)

- File system
  - Distributed file systems that store massive data and ensure – consistency, availability, and fault tolerance of data.
  - GFS is a notable example of distributed file system that supports large-scale file system, though it’s performance is limited in case of small files
  - Hadoop Distributed File System (HDFS) and Kosmosfs are other notable file systems, derived from the open source codes of GFS.

Industrial Data Management

- Incorporates data generated during
  - Manufacturing plants
  - Processing plants
- Management done in entire value chain
- Data availability is ensured
- Enables decision making process easier

Industrial Data Management: Advantages

- Production data of your plant is available
  - Raw material consumption
  - Production specifications
  - Energy Consumption
  - Plant utilization
  - Diagnostic information
- Enabling automated process

Data Management Using Hadoop

What is Hadoop

- Hadoop
  - Software framework for distributed processing of large datasets across large clusters of computers
  - Open-source implementation for Google File System (GFS) and MapReduce
  - MapReduce and Hadoop Distributed File System (HDFS) components originally derived respectively from Google's MapReduce and GFS.

Building Blocks of Hadoop

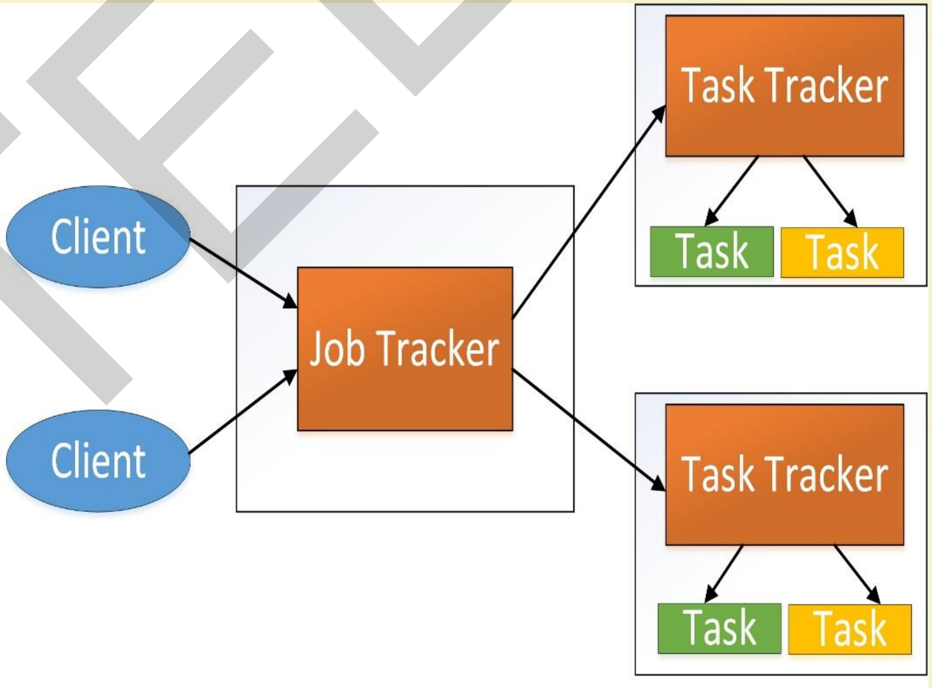
- Hadoop Common
  - A module containing the utilities that support the other Hadoop components
- Hadoop Distributed File System (HDFS)
  - Provides reliable data storage and access across the nodes
  - Rapid data transfer among the nodes
  - Fault tolerant

Building Blocks of Hadoop (Contd.)

- MapReduce
  - Framework for applications that process large amount of datasets in parallel
- Yet Another Resource Negotiator (YARN)
  - Next-generation MapReduce
  - Assigns CPU, memory and storage to applications running on a Hadoop cluster.

Hadoop Distributed File System (HDFS)

- Centralized node
  - Namenode
  - Maintains metadata info about files
- Distributed node
  - Datanode
  - Store the actual data
  - Files are divided into blocks
  - Each block is replicated

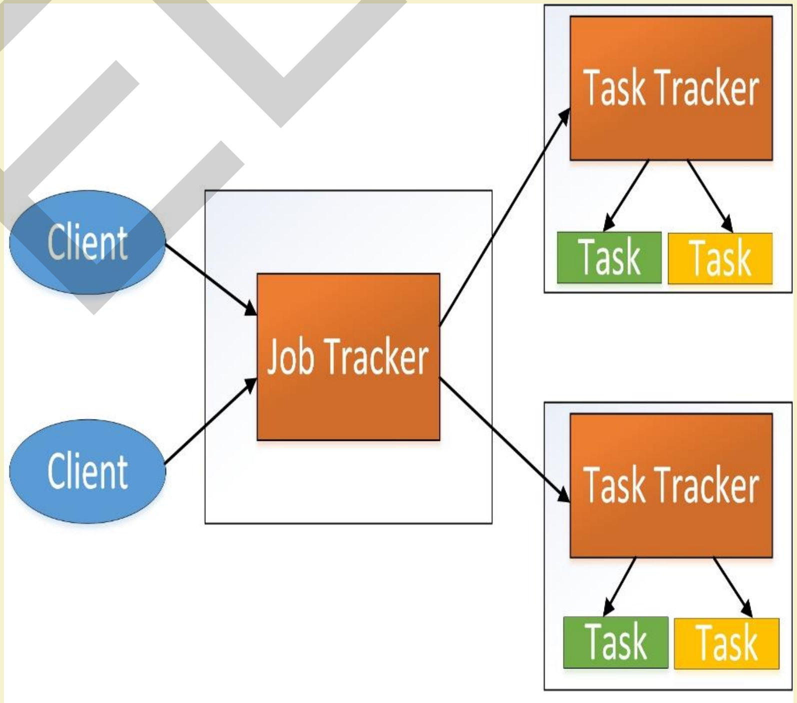


Name and Data Nodes

- Name Node
  - Stores filesystem metadata.
  - Maintains two in-memory tables, to map the datanodes to the blocks, and vice versa
- Data Node
  - Stores actual data
  - Can talk to each other to rebalance and replicate data
  - Update the namenode with the block information periodically
  - Before updating, datanodes verify the checksums

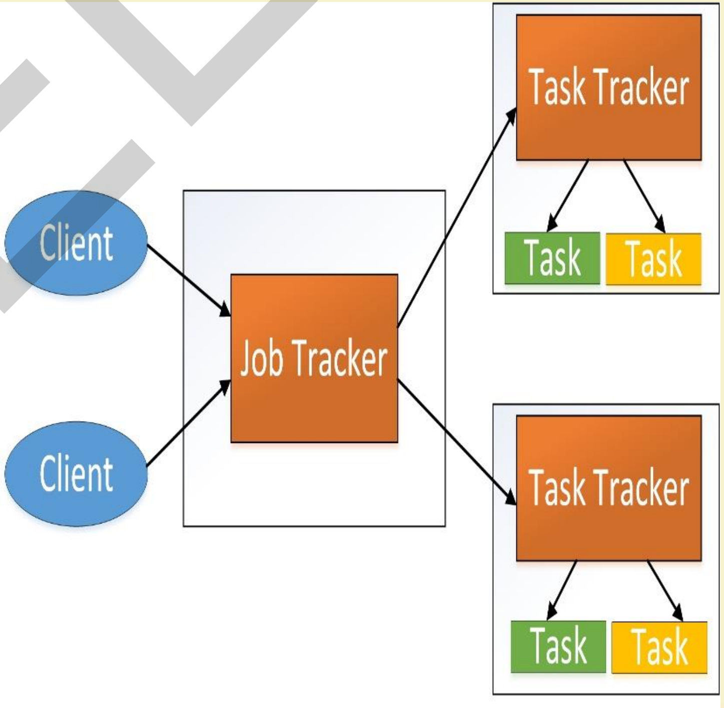
Job and Task Trackers

- Job Tracker
  - Runs with the Name Node
  - Receives the user's job
  - Decides on how many tasks will run (number of mappers)
  - Decides on where to run each mapper (concept of locality)
- Task Tracker
  - Runs on each Data Node
  - Receives the task from Job Tracker
  - Always in communication with the Job Tracker reporting progress



Hadoop Master/Slave Architecture

- Master
  - Executes operations like opening, closing, and renaming files and directories
  - Determines the mapping of blocks to Data Nodes
- Slave
  - Serves read and write requests from the file system's clients
  - Performs block creation, deletion, and replication as instructed by the Name Node

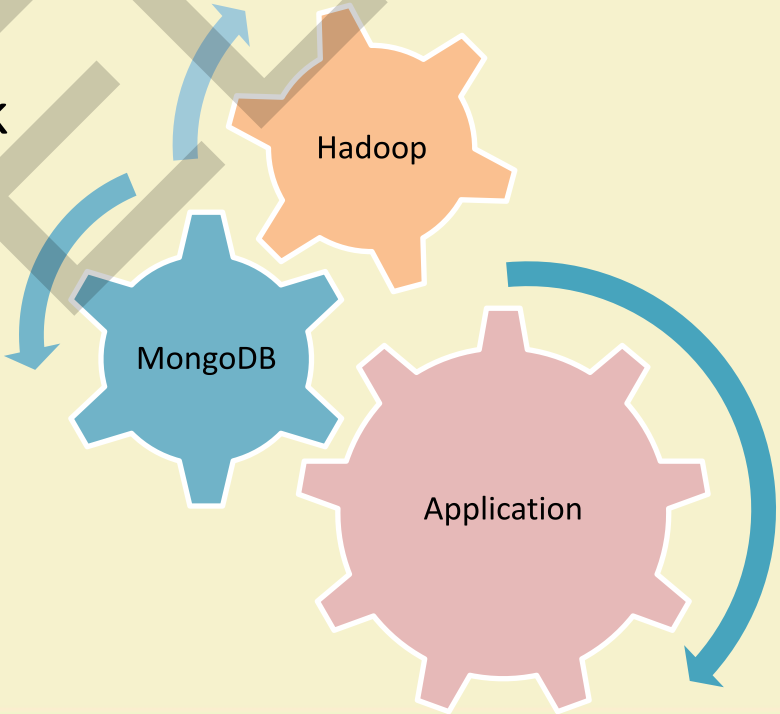


MongoDB in Data Management

- Uses Relational Database
- Ensures
  - Performance
  - Scalability
  - Availability
- Creates a similar view of data across the enterprise

MongoDB with Hadoop

- Hadoop adds a powerful framework to MongoDB for complex analytics
- Applications:
  - Batch Aggregation
  - Data Warehouse
  - ETL Data



References - I

1. R. Ahmed and G. Karypis, “Algorithms for Mining the Evolution of Conserved Relational States in Dynamic Networks,” Knowledge and Information Systems, vol. 33, no. 3, pp. 603-630, Dec. 2012.
2. M.H. Alam, J.W. Ha, and S.K. Lee, “Novel Approaches to Crawling Important Pages Early,” Knowledge and Information Systems, vol. 33, no. 3, pp 707-734, Dec. 2012.
3. S. Aral and D. Walker, “Identifying Influential and Susceptible Members of Social Networks,” Science, vol. 337, pp. 337-341, 2012.
4. A. Machanavajjhala and J.P. Reiter, “Big Privacy: Protecting Confidentiality in Big Data,” ACM Crossroads, vol. 19, no. 1, pp. 20-23, 2012.
5. S. Banerjee and N. Agarwal, “Analyzing Collective Behavior from Blogs Using Swarm Intelligence,” Knowledge and Information Systems, vol. 33, no. 3, pp. 523-547, Dec. 2012.
6. E. Birney, “The Making of ENCODE: Lessons for Big-Data Projects,” Nature, vol. 489, pp. 49-51, 2012.

References - II

7. S. Borgatti, A. Mehra, D. Brass, and G. Labianca, "Network Analysis in the Social Sciences," Science, vol. 323, pp. 892-895, 2009.
8. J. Bughin, M. Chui, and J. Manyika, Clouds, Big Data, and Smart Assets: Ten Tech-Enabled Business Trends to Watch. McKinSey Quarterly, 2010.
9. D. Centola, "The Spread of Behavior in an Online Social Network Experiment," Science, vol. 329, pp. 1194-1197, 2010.
10. http://hadoop.apache.org/

Thank You!!

IIoT Analytics and Data Management: Data Center Networks

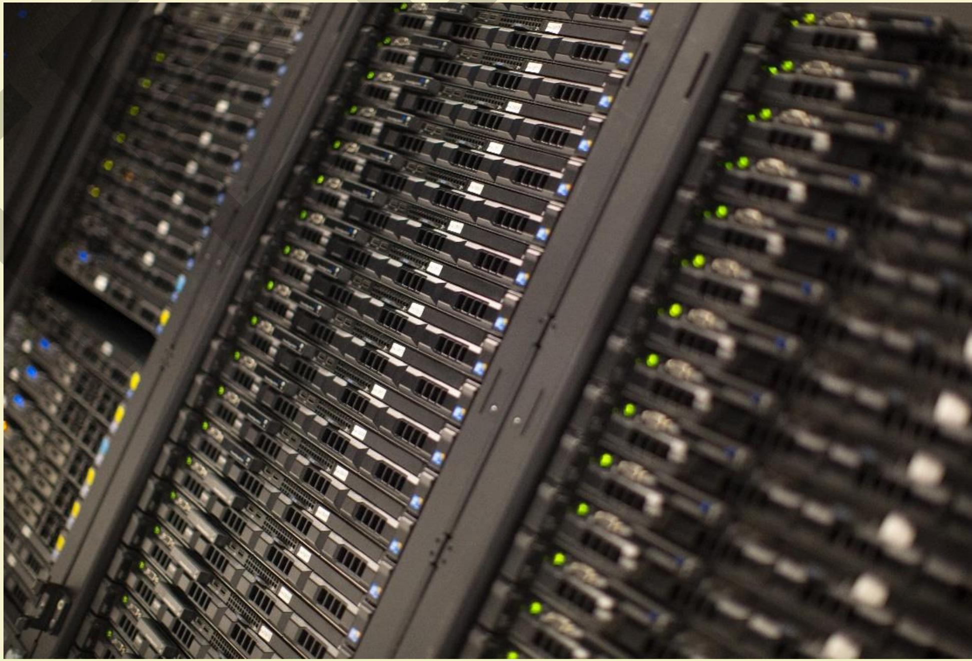
Dr. Sudip Misra
Professor
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Email: smisra@sit.iitkgp.ernet.in
Website: http://cse.iitkgp.ac.in/~smisra/
Research Lab: cse.iitkgp.ac.in/~smisra/swan/

What is Data Center Networks

- Data Center
  - Composed of networked computers and storage
  - Core of an organization's information system
- Data center networks
  - Interconnects the different data center resources such as computational, storage, network entities
  - Accommodates different data centers having varying dataload

Data Center (Example)

- Wikimedia Foundation



Source: Wikimedia Foundation Servers 2015-90.jpg, VGrigas (WMF), Published date: 21 July 2015, Online: https://commons.wikimedia.org/wiki/File:Wikimedia_Foundation_Servers_2015-90.jpg

Data Center Network: Properties

- Stable
- Secure
- Reliable
- Supports networking requirements
- Scalable
- Agility (any service on any server at any time)

Data Center Network: Requirements

- VM migration without changing IP address
- No need to configure switch before deployment
- Path should be available among the end-users to communicate
- Fast detection of failure
- Efficient repair of failure

Data Center Applications

- Outward facing applications
  - Serving web pages to users
- Internal computational applications
  - MapReduce for web indexing
- Running multiple concurrent services

Data Center Network: Topology

- Three-tier DCN
- Fat Tree DCN
- Dcell
- BCube

Data Center Network: Topology (Contd.)

- Three-Tier DCN
  - Multi-rooted tree based network topology
  - Three layers of network switches
    - Edge
    - Aggregate
    - Core
  - Disadvantages:
    - Scalability, fault tolerance, energy efficiency, and cross-sectional bandwidth

Data Center Network: Topology (Contd.)

- Fat-Tree DCN
  - Inter connects K-ary Fat tree
  - Three-tier topology
    - Edge, Aggregation, Core
  - Pod at edge tier consists of $(k/2)^2$ servers and (k/2) k-port switches
  - Each edge switch connects to (k/2) servers and (k/2) aggregation switches
  - Each aggregation switch connects to (k/2) edge and (k/2) core switches
  - $(k/2)^2$ core switches, each of which connects to k pods

Data Center Network: Topology (Contd.)

- DCell
  - Uses a recursively-defined structure to interconnect server
  - Server is connected to several other servers and a mini-switch via communication links
  - Low-level DCells form a fully-connected graph
  - Fault tolerant
  - No single point of failure

Data Center Network: Topology (Contd.)

- BCube
  - Server-centric approach, rather than the switch-oriented practice
  - Places intelligence on modular data center (MDC) servers
  - Provides multiple parallel short paths between any pair of server
  - Constructs edge-disjoint complete graphs
  - Forms multiple edge-disjoint server spanning tree
  - Runs a source routing protocol called BSR (Bcube Source Routing)

Data Center Network: Technology

- Networking equipment
  - Routers
  - Switches
  - Modems
- Network cabling
  - LAN/WAN
  - Network interface cabling

Data Center Network: Technology (Contd.)

- Network addressing scheme
  - IPV4
  - IPV6
- Network security
  - Security protocols or encryption algorithms
  - Firewalls
- Internet connectivity
  - Satellite, wireless, optical

Data Center Network: Challenges

- Scalability
- Poor server-to-server Connectivity
- Static resource assignment
- Resource Fragmentation
- Fault-tolerance

Data Center and IIoT: Challenges

- Data
- Security
- Consumer Privacy
- High Availability
- Storage Management
- Data Center Network

Data and IIoT: Challenges

- Generates a substantial amount of data
- Continuously learn about the end-user and industrial appliances
- Storage
  - Consumer Driven
  - Enterprise driven

Security and IIoT: Challenges

- Connects a large number of assets or device
- Communicate automatically
- Increase in digitization and automation of devices
- Devices are spread across different areas
- Absence of a secure and properly encrypted network

Consumer Privacy and IIoT: Challenges

- Presence of several IoT connected things
- Vast amounts of data
- Information on users' personal use of devices
- Personal information generated by the devices serves as the key to bringing improved services
- Improve management of IoT devices at industries

High Availability and IIoT: Challenges

- Innumerable devices are connected
- Generated big data
- Increase in the complexity of security management
- Impact due to security challenges
- Real-time business process
- Personal data safety

Storage Management and IIoT: Challenges

- Increase in demand of storage capacity
- Large amounts of data generated by connected devices
- Cost efficient storage for IoT devices

Data Center Network and IIoT: Challenges

- Support for bandwidth requirements
- Drastic change in bandwidth pattern
- Bulk amount of small messages having sensor data
- Requirement for increase in inbound data center bandwidth

Software-Defined Data Center for IIoT

- Software defined data center
  - Virtualized data storage
  - Data center as a service
- Abstracted from hardware
  - Deployment
  - Operation
  - Provisioning
  - Configuration

Software-Defined Data Center: Components

- Network virtualization
- Storage virtualization
- Server virtualization
- Business logic layer

Software-Defined Data Center: Advantages

- Separation of control and data panel
- Agility
- Elasticity
- Scalability
- Cloud Computing
- Programmable infrastructural and workload management

References - I

[1] Md. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in Proceedings of the ACM SIGCOMM conference on Data communication (SIGCOMM '08), New York, NY, USA, 2008, pp. 63-74.

[2] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," SIGCOMM Comput. Commun. Rev. 39, 1, December 2008, pp. 68-73.

[3] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," in Proceedings of the ACM SIGCOMM conference on Data communication (SIGCOMM '08), New York, NY, USA, 2008, pp. 75-86.

[4] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a high performance, server-centric network architecture for modular data centers," in Proceedings of the ACM SIGCOMM conference on Data communication (SIGCOMM '09), New York, NY, USA, 2009, 63-74.

[5] M. F. Bari et al., "Data Center Network Virtualization: A Survey," IEEE Communications Surveys & Tutorials, vol. 15, no. 2, pp. 909-928, Second Quarter 2013.

References - II

[6] M. S. Hossain and G. Muhammad, "Cloud-assisted Industrial Internet of Things (IIoT) – Enabled framework for health monitoring," Computer Networks, Vol. 101, 2016, pp. 192-202.

[7] H. Ning and Z. Wang, "Future Internet of Things Architecture: Like Mankind Neural System or Social Organization Framework?," IEEE Communications Letters, Vol. 15, No. 4, April 2011, pp. 461-463.

[8] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," Business Horizons, Vol. 58, No. 4, 2015, pp. 431-440

[9] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," IEEE Communications Magazine, vol. 51, no. 11, pp. 24-31, November 2013.

[10] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: experience with a globally-deployed software defined wan," in Proceedings of the ACM SIGCOMM conference on SIGCOMM (SIGCOMM '13), New York, NY, USA, 2013, pp. 3-14.

[11] D. Li, Y. Shang and C. Chen, "Software defined green data center network with exclusive routing," IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, Toronto, ON, 2014, pp. 1743-1751.

Thank You!!

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

NPTEL

Advanced Technologies: Software-Defined Networking (SDN) in IIoT – Part 1

Dr. Sudip Misra

Professor

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Email: smisra@sit.iitkgp.ernet.in

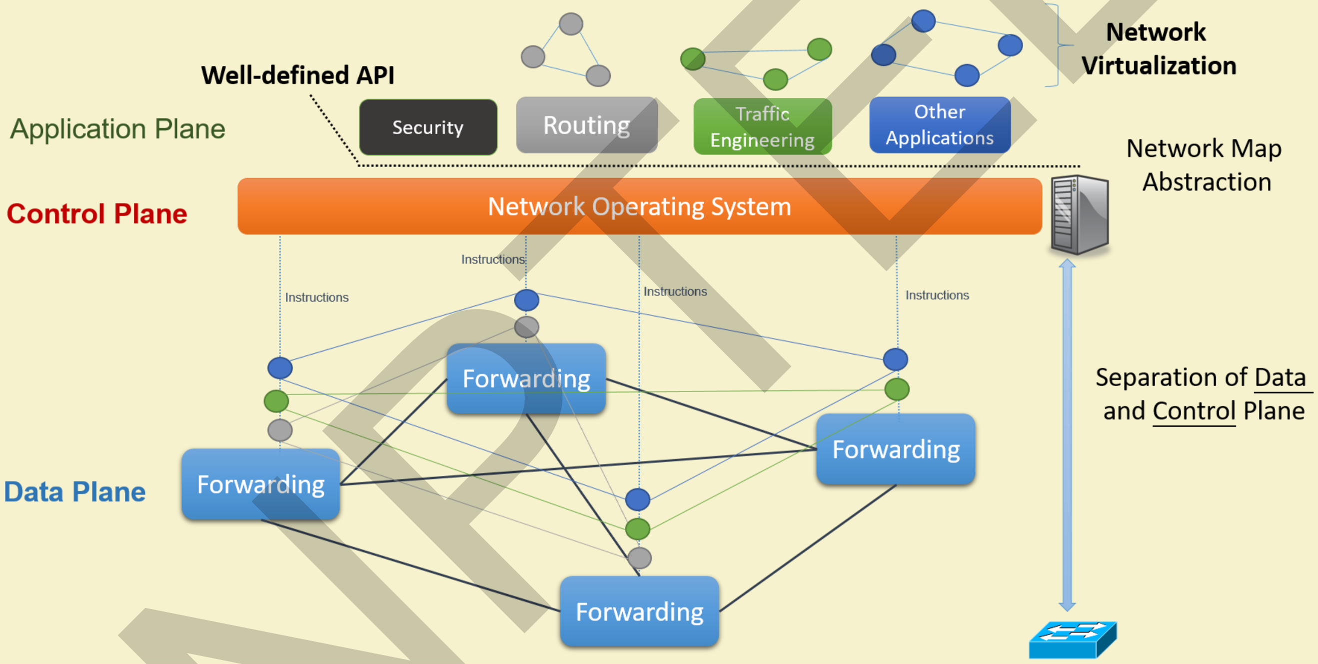
Website: http://cse.iitkgp.ac.in/~smisra/

Research Lab: cse.iitkgp.ac.in/~smisra/swan/

Software-Defined Network (SDN)

- What is SDN?
  - Restructuring the current network infrastructure for improved network management.
  - It is not a new technology – rather reshaping the current network architecture.
  - Control and data planes are decoupled from the traditional forwarding devices.

SDN Architecture



SDN Components/Attributes

- Application programming interfaces (APIs)
- Controller
- Forwarding devices
- Protocol – OpenFlow
- Applications

SDN Aspects

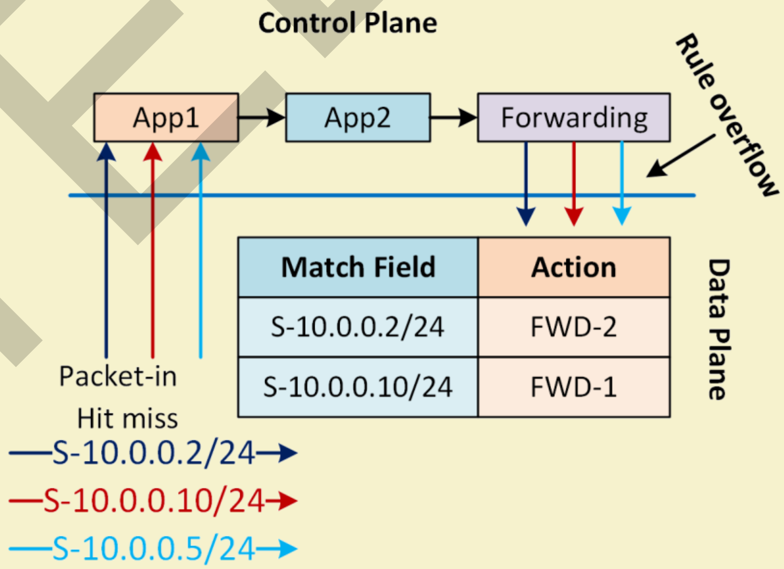
- Rule Placement
- Controller Placement
- Security

Rule Placement

- Forwarding devices forward an incoming traffic based on the control logic defined by the SDN controller.
- The control logic is placed at the devices in the form of flow-rule.
- Ternary content addressable memory (TCAM) available at the devices is used to place the flow-rules.
- TCAM is limited – limited number of flow-rules can be placed.

Rule Placement (contd.)

- Flow-rule for first two flows are inserted.
- Rule for third incoming flow cannot be inserted due to rule capacity constraint.
- How to accommodate the new flow?
  - Existing rule may be deleted
  - Two rules may be combined (wildcard) to make them one rule



Controller Placement

- How many controllers required?
- What should be there placement – flat, hierarchical, etc.
- What about fault-tolerance – backup controller?

Security

- Firewall
- DoS attack
- Reliable and secure connection between SDN controller and forwarding devices
  - Currently, TCP with TLS is used for communication between controller and forwarding devices.

SDN Applications I

- Network management – backbone Internet
- Traffic Engineering (Katta et al., 2016)
- Load Balancing (Qiao et al., 2016)
- Dynamic access control between user and access points (Suresh et al., 2012)
- Mobility Management (Li et al., 2014; Bera et al., 2016)

SDN Applications II

- WSN Management (Galluccio et al., 2015; Bera et al., 2016)
- IoT Applications (Bera et al., 2017)
- IIoT Applications (Wan et al., 2016)

Software-Defined IIoT (SDIIoT)

- Challenges/Requirements in IIoT network:
  - Network Segmentation
  - Policy-based data forwarding
  - Remote control of devices' functionalities
  - Security

Network Segmentation

- Data from IIoT system is typically follows UDP service.
- Streaming the UDP data over TCP/IP may reduce network performance.
- If want to use the same/common network for all applications, network architecture and forwarding policies need to be changed.

- SDN is capable of address such issues by separating control and data planes from the traditional forwarding devices.

Policy-based Data Forwarding

- Several sensors/actuators would be placed to monitor/actuate real-time status of industrial equipment.
- Forwarding policies may need to change dynamically depending on real-time situation.
- For example, temperature data may have higher priority compared to humidity, and vice-versa, in different time periods. How to meet such requirements dynamically?
- Rule-based forwarding policies in SDN would be capable of meeting such requirements of IIoT.

Remote Control of Devices' Functionalities

- A device with multiple sensors may be planted in an industrial component to monitor different parameters simultaneously or according to requirements.
- The system should be capable of controlling the sensor-device's functionality remotely to meet requirements.
- Software-defined approach is capable of achieving such requirements.

Security

- Securing the network and device is another important aspect.
- Flow-based forwarding in SDN is capable of preventing DoS attacks.
- Customized middleware is also useful for improved security in IIoT network.

SDIIoT Advantages

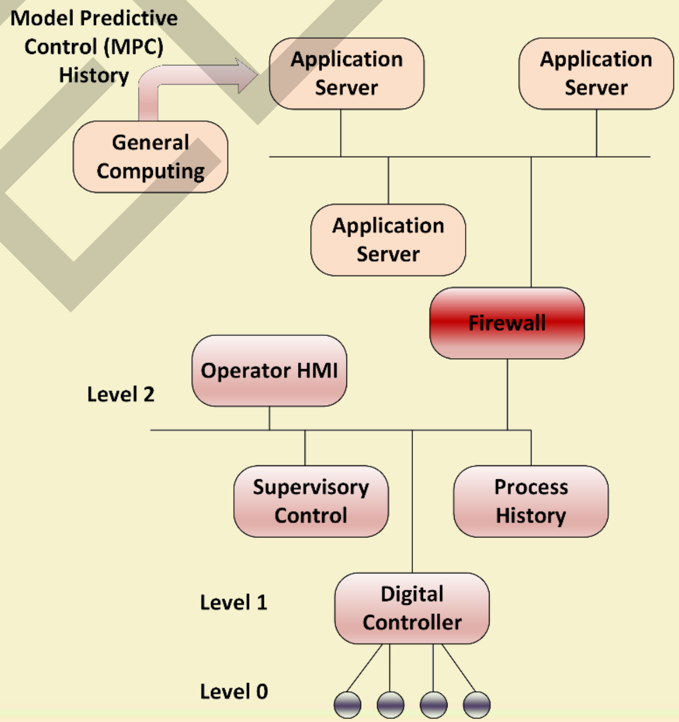
- Low-latency virtualization of VMs
  - Dynamic capacity adjustment based on demand
  - Easy migration of VMs using software-defined policies
- Deterministic networking
  - Logically centralized view of the network
  - Rule-based forwarding enables deterministic forwarding of traffic over network – so that events are processed in order

- High availability
  - Fault-tolerance feature of SDN would help IIoT system to enable new servers or software to deal with faults
  - Carrier grade telecommunication NFV is capable of meeting such requirements
- Robust security
  - Centralized view of the devices and events provides improved security
  - Each component of IIoT system would be monitored – which will help us to prevent unwanted access of the system

- Up-to-date applications
  - The open architecture of devices enables administrators to run up-to-date applications
  - Cost-effective, reliable, and secure management is possible by using the up-to-date applications

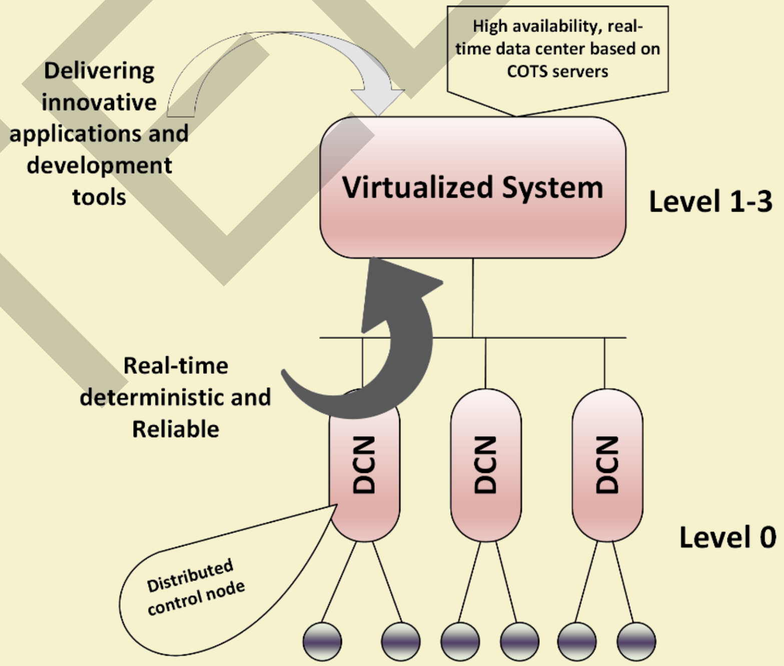
Current Practice: Automation

- Level 0 – Sensors
- Level 1 – Digital controllers
- Level 2 – Supervisory control



Software-Defined Automation

- Virtualized platform
- Dynamic, real-time control of systems



References

- N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "CacheFlow: Dependency-Aware Rule-Caching for Software-Defined Networks," in *Proc. of the Symposium on SDN Research (SOSR)*, no. 6, CA, USA, Mar. 2016.
- S. Qiao, C. Hu, X. Guan, and J. Zou, "Taming the Flow Table Overflow in OpenFlow Switch," in *Proceedings of the ACM SIGCOMM*, Florianopolis, Brazil, Aug. 2016, pp. 591–592.
- L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise WLANS with Odin," in *Proc. of the ACM Workshop on HotSDN*, Helsinki, Finland, Aug. 2012, pp. 115–120.
- H. Li, P. Li, and S. Guo, "MoRule: Optimized rule placement for mobile users in SDN-enabled access networks," in *Proc. of the IEEE GLOBECOM*, TX, Dec. 2014, pp. 4953–4958.
- S. Bera, S. Misra, and M.S. Obaidat, "Mobility-Aware Flow-Table Implementation in Software-Defined IoT", *in Proc. of the IEEE GLOBECOM*, 2016.

References

- L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, “SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks,” in *Proc. of the IEEE INFOCOM*, Kowloon, Apr.-May 2015, pp. 513–521.
- S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, "Soft-WSN: Software-Defined WSN Management System for IoT Applications", *IEEE Systems Journal*, 2016.
- S. Bera, S. Misra, and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey", *IEEE Internet of Things Journal*, 2017.
- J. Wan, S. Tang, Z. Shu, D. Li, S. Wang, M. Imran, and A. V. Vasilakos, “Software-defined industrial Internet of Things in the context of industry 4.0,” *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7373–7380, Oct. 2016.
- Available Online: http://blogs.windriver.com/wind_river_blog/2016/11/software-defined-infrastructure-in-industrial-iot-how-it-works.html, Accessed on August 12, 2018.

Thank You!!