دومین کارگاه یادگیری ماشینی در فیزیک: کاربردها در ماده چگال

مبانی و مفاهیم یادگیری ماشینی و بیگ دیتا

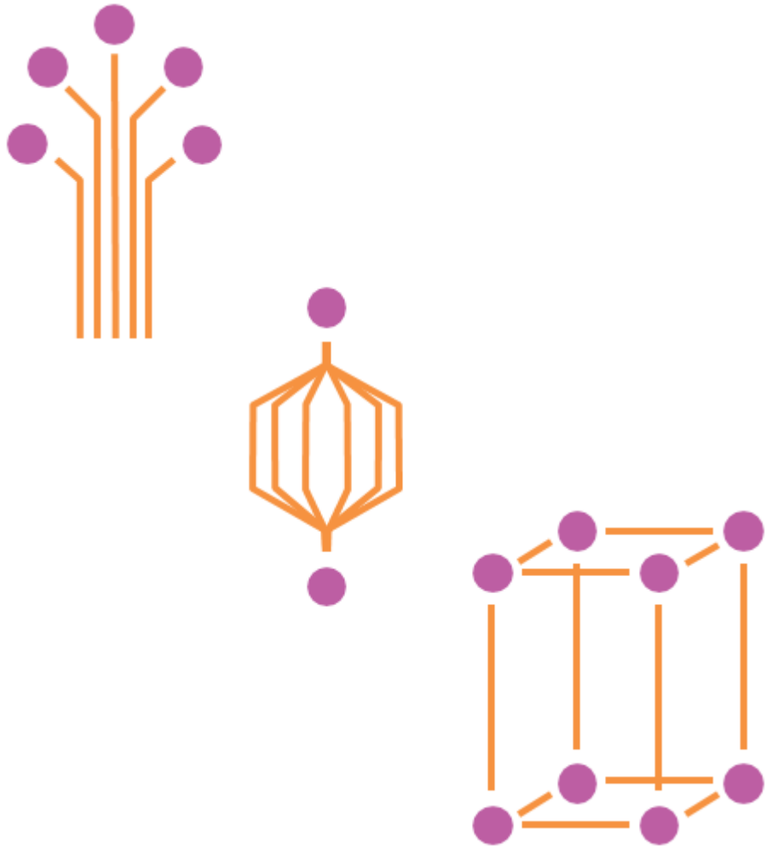
۱۱-۱۳ مهرماه ۱۳۹۷
دانشگاه تهران

مبانی یادگیری ماشینی

Amin Nezarat (Ph.D.)
Assistant Professor at Payame Noor University
aminnezarat@gmail.com www.astek.ir - www.hpclab.ir

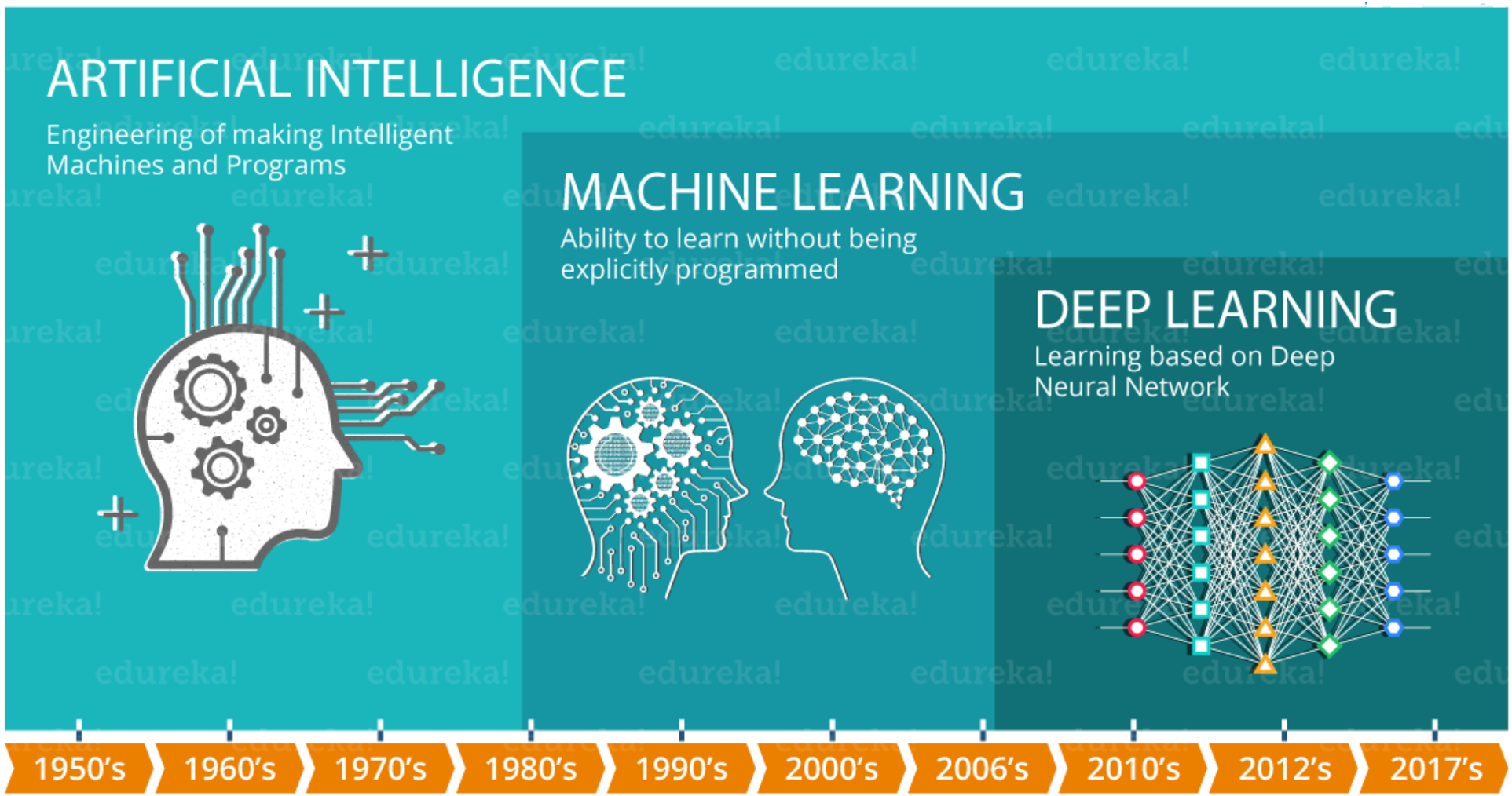
عناوین دوره

۱. مبانی یادگیری ماشینی
۲. بیگ دیتا و مفاهیم
۳. افزایش کارایی الگوریتمهای یادگیری ماشینی
۴.



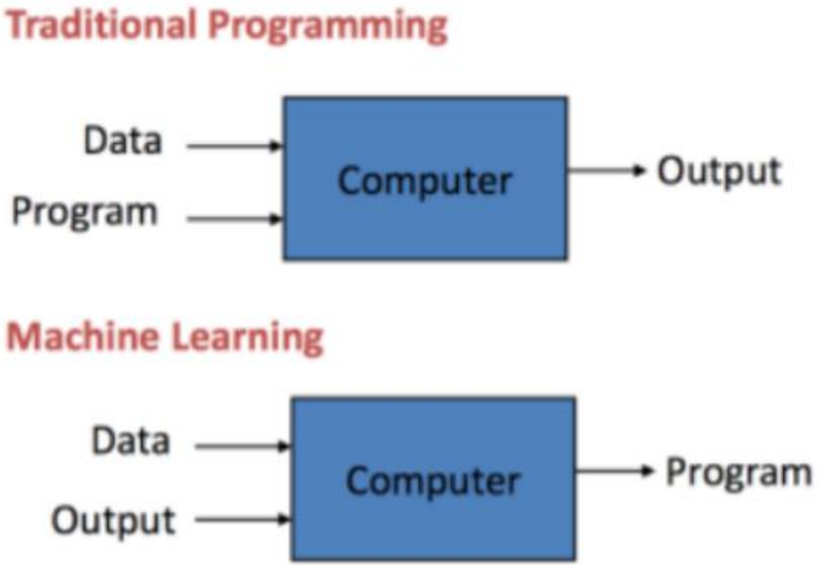
مبانی یادگیری ماشینی

چرخه عمر یادگیری ماشینی



❖تعریف یادگیری ماشینی

- **Tom Mitchell (1998):** A computer program is said to learn from experience with respect to some task and some performance measure , if its performance on , as measured by , improves with experience.
- **Ethem Alpaydın (2010):** Machine learning is programming computers to optimize a performance criterion using example data or past experience.
- Traditional programming versus machine learning:



❖چرا یادگیری؟

- چرا ماشین را برنامه نویسی نکنیم؟
  - بعضی کارها را بدرستی نمیتوان توصیف نمود. در صورتیکه ممکن است آنها را بتوان بصورت مثالهای (ورودی/خروجی) معین نمود.
  - ممکن است در خیل عظیمی از داده اطلاعات مهمی نهفته باشد که بشر قادر به تشخیص آن نباشد (داده کاوی).
  - ممکن است موقع طراحی یک سیستم تمامی ویژگیهای آن شناخته شده نباشد در حالیکه ماشین میتواند حین کار آنها را یاد بگیرد.
  - ممکن است محیط در طول زمان تغییر کند. ماشین میتواند با یادگیری این تغییرات خود را با آنها وفق دهد.

❖چرا یادگیری؟

- در عمل نوشتن برنامه ای برای تشخیص یک صورت میتواند کار مشکلی باشد. زیرا تعریف دقیقی برای آن وجود ندارد و حتی در صورت وجود داشتن نوشتن برنامه ای بر اساس آن کار سختی است.

- در نتیجه بجای نوشتن یک برنامه با دست، میتوان با تهیه مقدار زیادی نمونه صحیح و اعمال آن به یک الگوریتم یادگیری ماشین برنامه ای تولید کنیم که کار مورد نظر را انجام دهد.

- این برنامه بسیار با آنچه که قرار بود با دست نوشته شود متفاوت خواهد بود. اگر این برنامه درست تهیه شده باشد میتواند برای نمونه هائی که تاکنون ندیده است نیز خروجی مورد نظر را تولید کند.

❖چرا یادگیری؟

- در سالهای اخیر پیشرفتهای زیادی در الگوریتم ها و تئوری های مربوطه بوجود آمده و زمینه های تحقیقاتی جدید زیادی پدید آمده اند.
- داده های آزمایشی زیادی بصورت Online بوجود آمده اند.
- کامپیوتر ها قدرت محاسباتی زیادی بدست آورده اند
- جنیه های عملی با کاربردهای صنعتی بوجود آمده اند. (در زمینه پردازش گفتار برنامه های مبتنی بر یادگیری از همه روشهای دیگر پیشی گرفته اند)

یادگیری مدل

- در یادگیری ماشین با استفاده از تئوری اطلاعات مدلهای ریاضی ساخته میشود که میتوانند برای استنتاج استفاده شوند.
- مدل ممکن است پیشگویانه (Predictive) باشد که برای پیش بینی موارد جدید بکار می روند.
- مدل ممکن است توصیفی باشد (descriptive) که دانش آن از داده یادگرفته میشود.
- البته مدل می تواند هر دو خاصیت فوق را داشته باشد.

یادگیری انسان و ماشین

- یادگیری انسان چگونه است؟
  - انسان از طریق تعامل با محیط بیرونی یاد میگیرد
  - یک عامل باید وجود داشته باشد تا یادگیری را شروع کند

- یادگیری ماشین چگونه اشت؟
  - از طریق نوشتن برنامه میتوان به ماشین گفت که چه باید بکند.
  - از طریق نمایش مثالهای متعدد میتوان ماشین را وادار به یادگیری نمود.
  - ماشین میتواند از طریق تجربه محیط واقعی یاد بگیرید.
  - در حالتیکه مثالها مشخص نیستند و خبره ای وجود ندارد ماشین میتواند از طریق مشاهده یادبگیرد.

❖تکنیکهای مختلف یادگیری

- **یادگیری استنتاجی (inductive)**

که یادگیری بر مبنای مثالهای متعدد انجام میشود. مثل درخت های تصمیم

- **یادگیری Connectionist**

که یادگیری بر مبنای مدل مغز بشر صورت میپذیرد. مثل شبکه های عصبی مصنوعی

- **یادگیری Bayesian**

که فرضیه های مختلفی در مورد داده ارائه میشود.

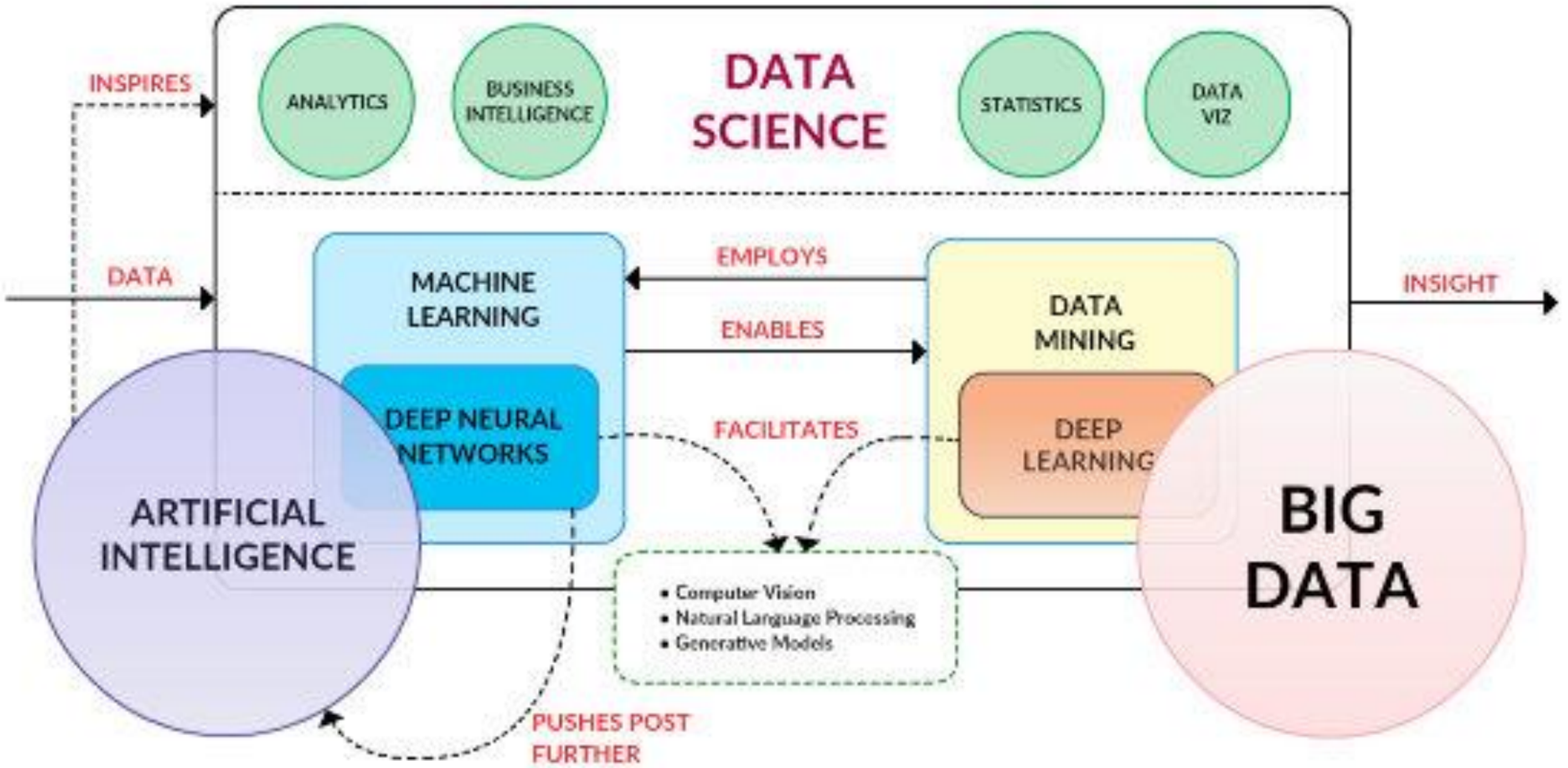
- **یادگیری Reinforcement**

که از سنسورها و تجربه در محیط استفاده میشود.

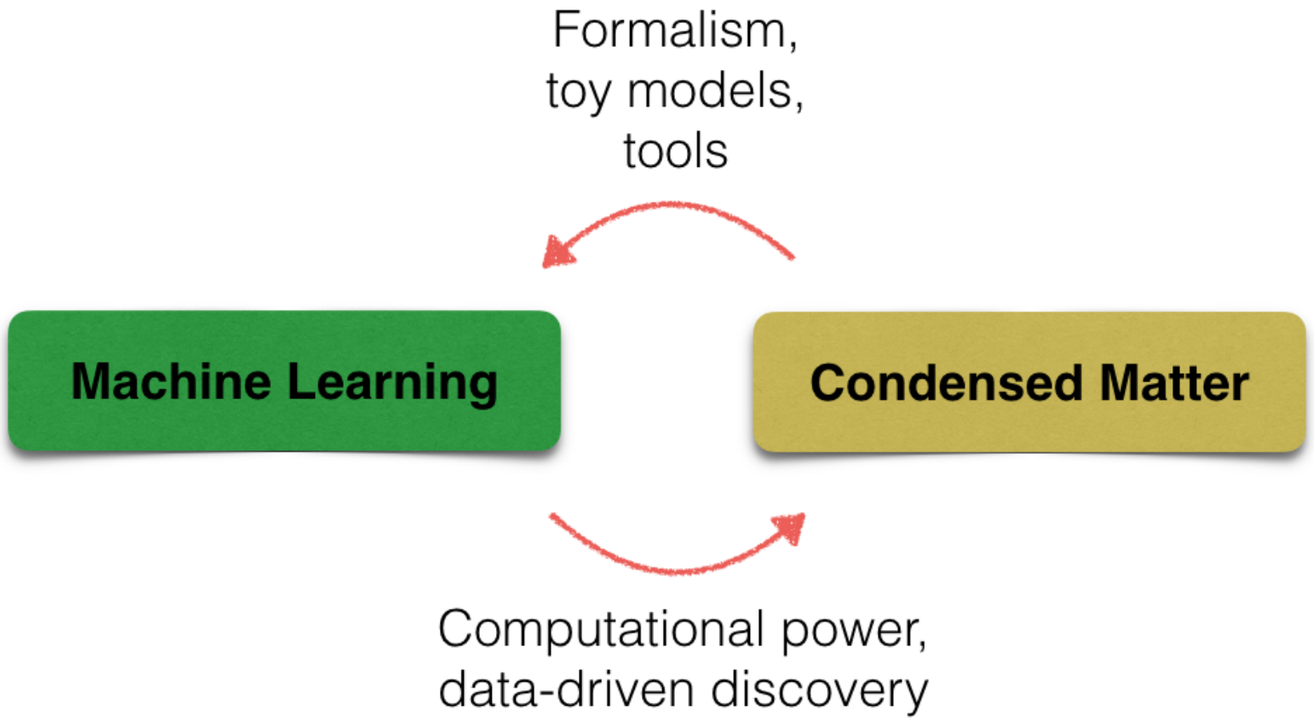
- **یادگیری Evolutionary**

مثل الگوریتم ژنتیک

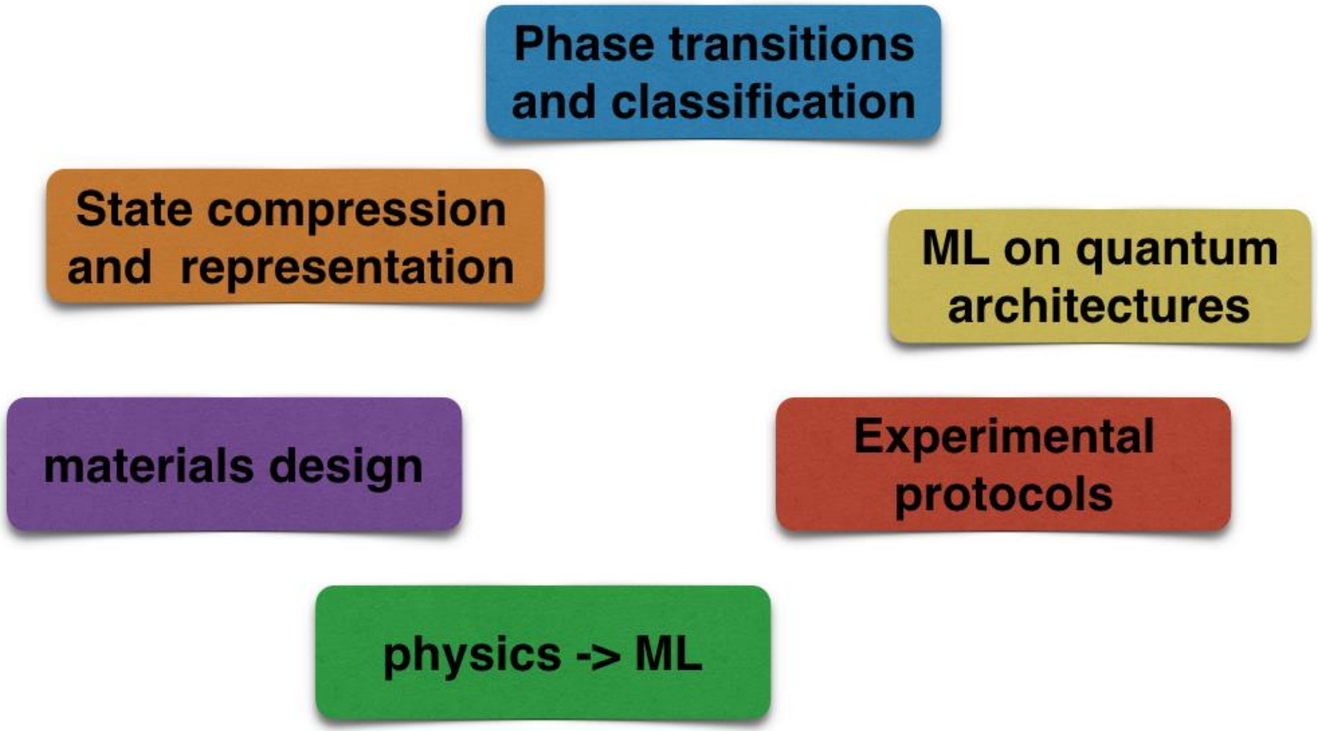
یادگیری ماشینی و بیگ دیتا



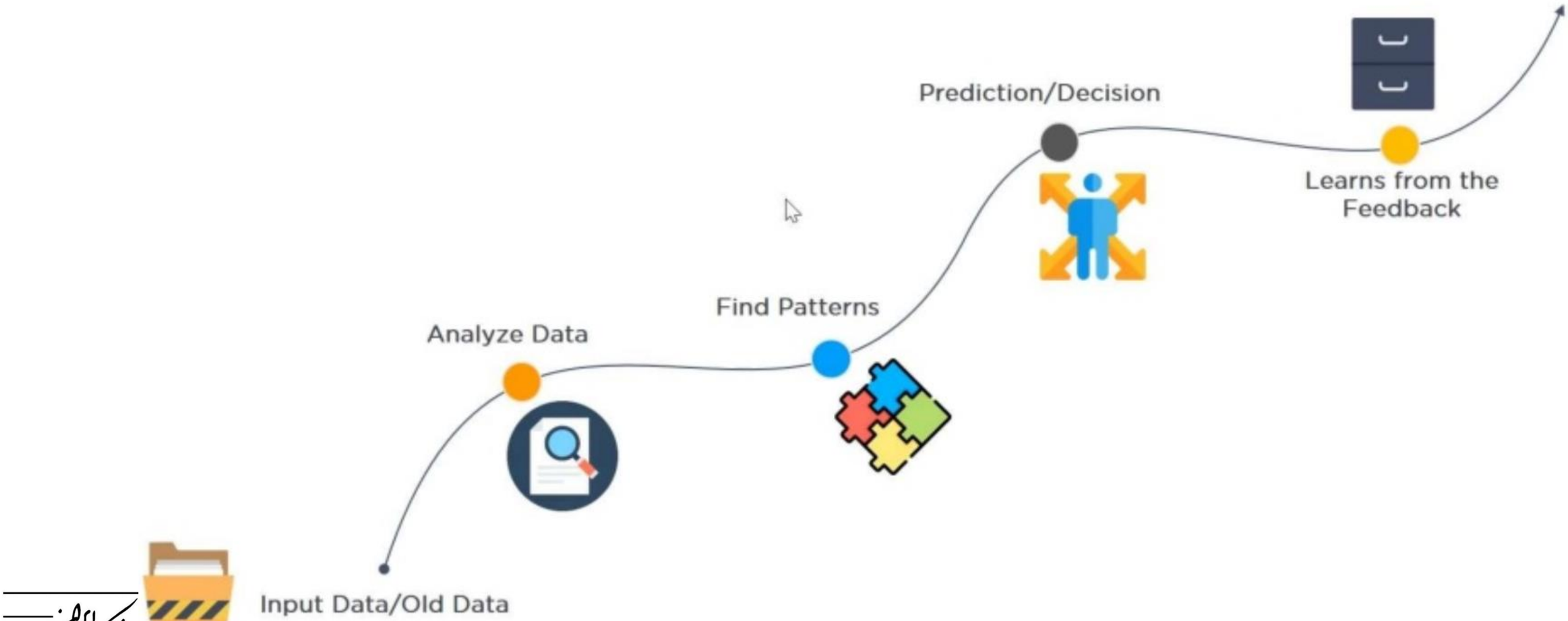
❖چرا یادگیری ماشینی



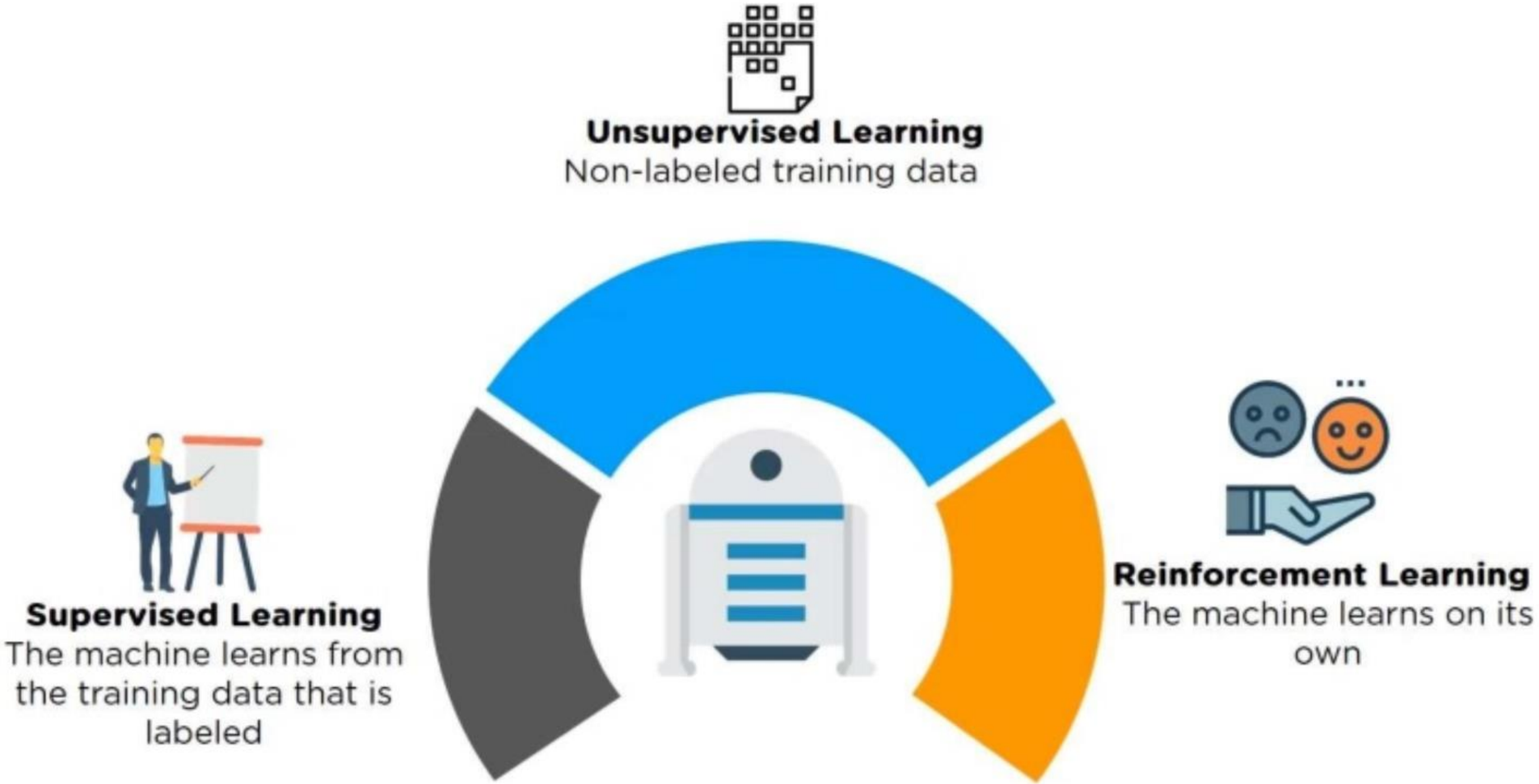
❖چه کاربردهایی دارد؟



❖چرخه ML


Prediction/Decision
Learns from the Feedback
Find Patterns
Analyze Data
Input Data/Old Data

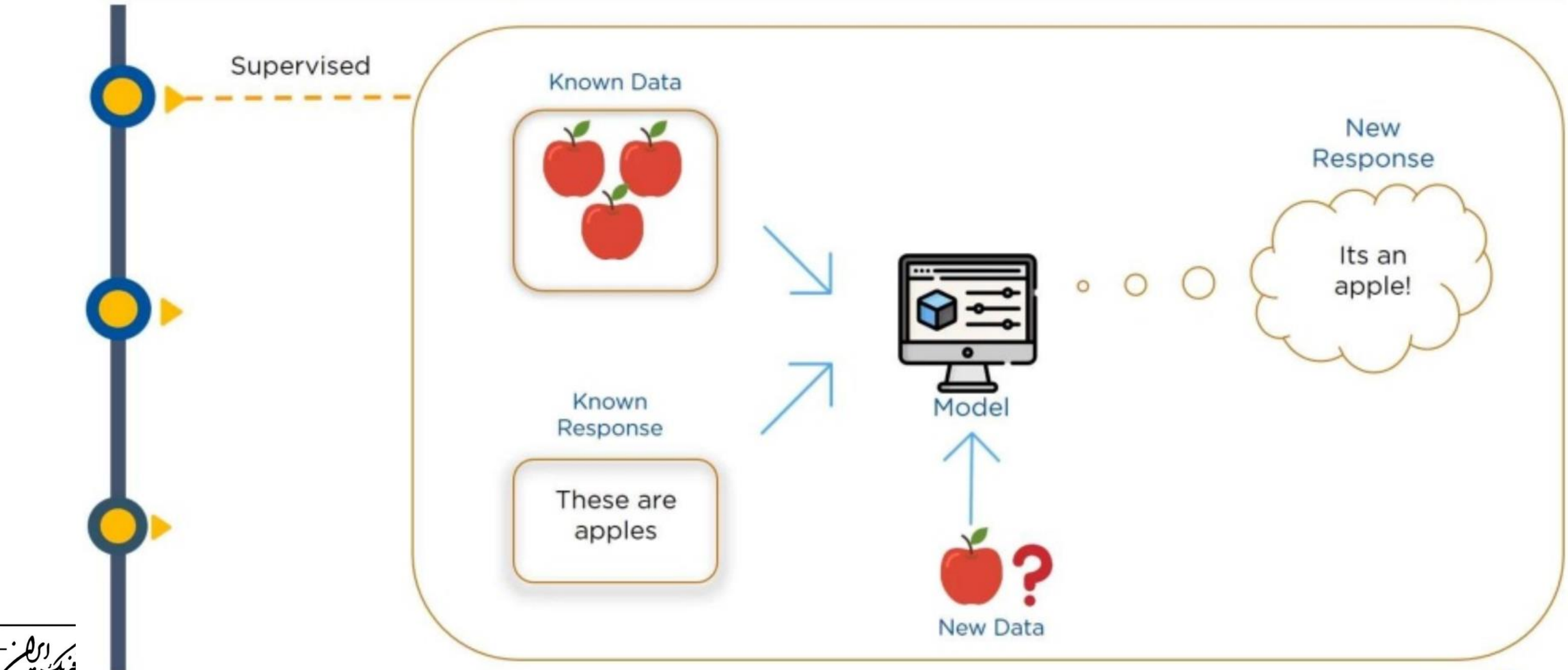
❖ انواع الگوریتمهای ML



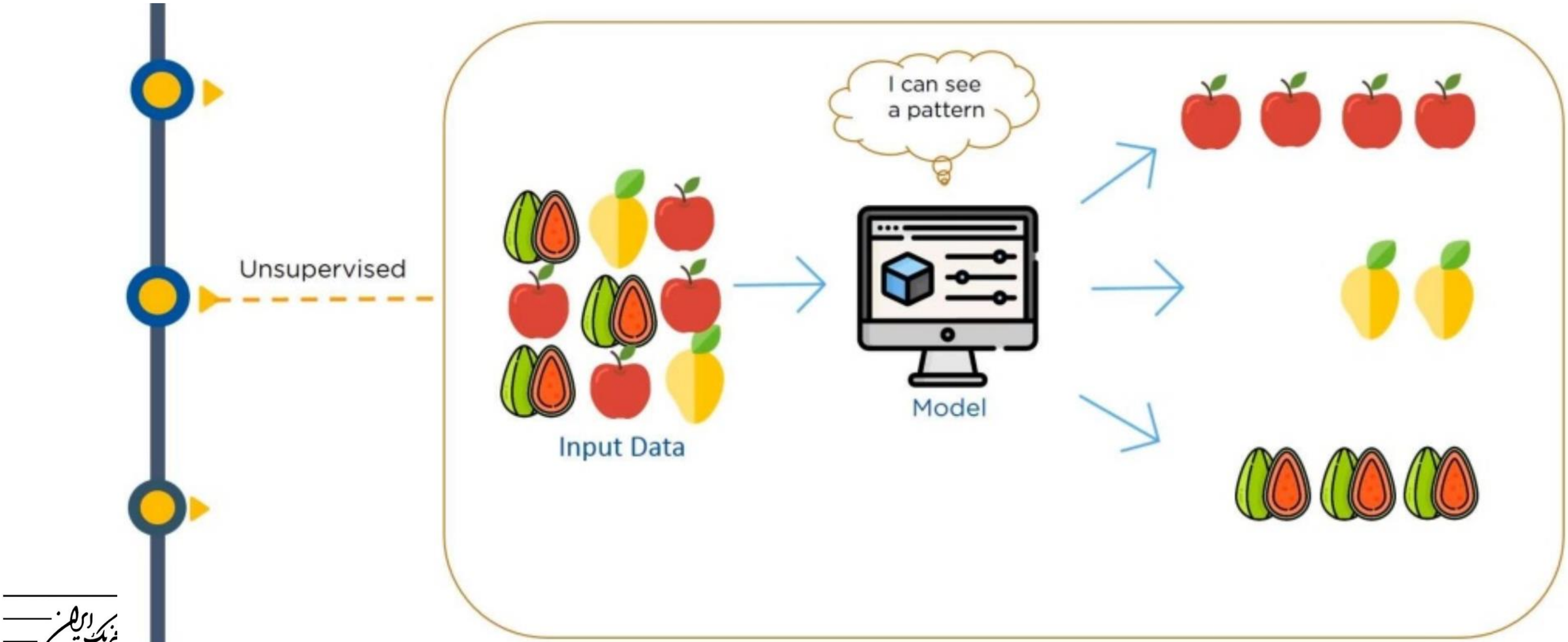
❖ انواع الگوریتمهای ML

- **یادگیری با ناظر:**
  یک مجموعه از مثالهای یادگیری وجود دارد بازای هر ورودی، مقدار خروجی و یا تابع مربوطه نیز مشخص است. هدف سیستم یادگیر بدست آوردن فرضیه ای است که تابع و یا رابطه بین ورودی و یا خروجی را حدس بزند
- **یادگیری بدون ناظر:**
  یک مجموعه از مثالهای یادگیری وجود دارد که در آن فقط مقدار ورودی ها مشخص است و اطلاعاتی در مورد خروجی صحیح در دست نیست. یادگیری بدون ناظر برای دسته بندی ورودیها و یا پیش بینی مقدار بعدی بر اساس موقعیت فعلی بکار میرود.
- **یادگیری تقویتی:**
  مثالها بصورت ورودی/خروجی نیستند بلکه بصورت وضعیت/پاداش هستند که یادگیر در وضعیت های مختلف عملیات مختلفی را انجام داده و پاداشهای متفاوتی دریافت و بر اساس مجموع پاداش های دریافتی عمل متناسب با هر وضعیت را یاد میگیرد.
- **یادگیری نیمه نظارتی:**
  مثالها طوری هستند که برای تعداد کمی از آنها مقدار خروجی موجود است اما برای مثالهای زیادی مقدار خروجی مشخص نیست.

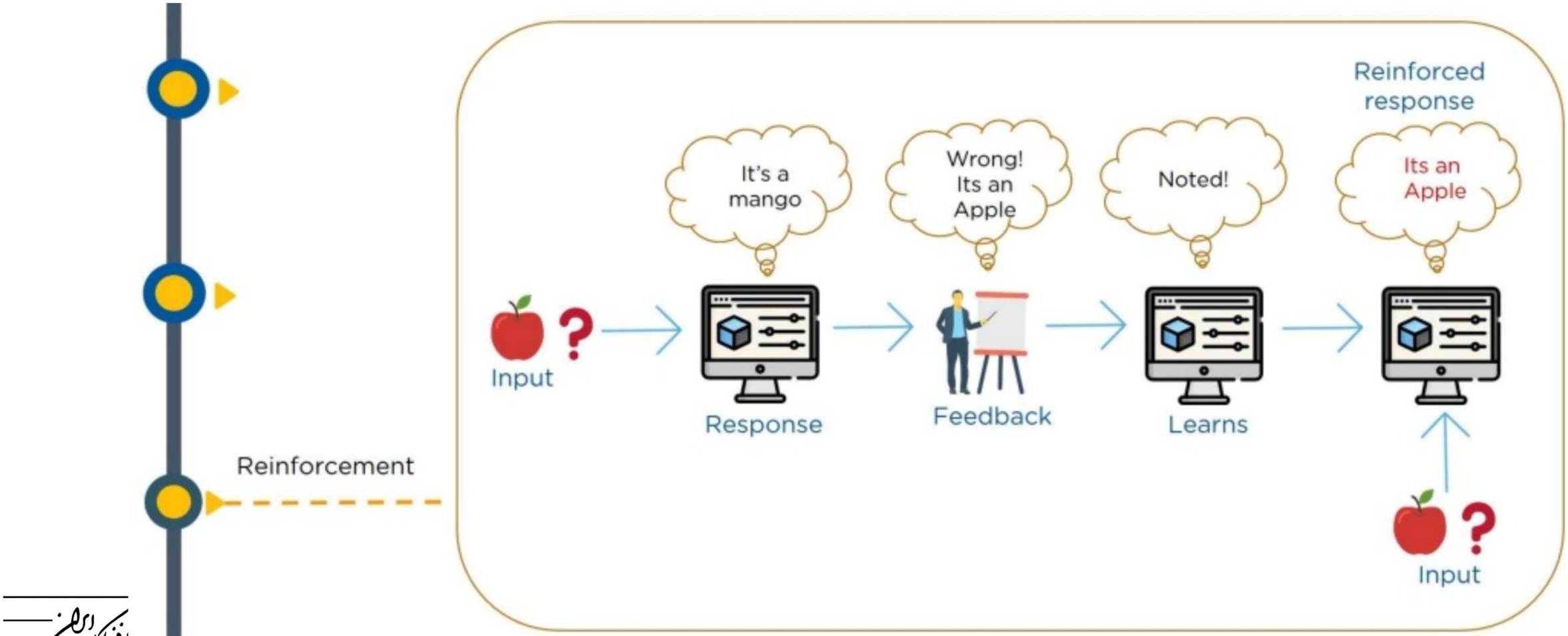
❖ الگوریتمهای با ناظر



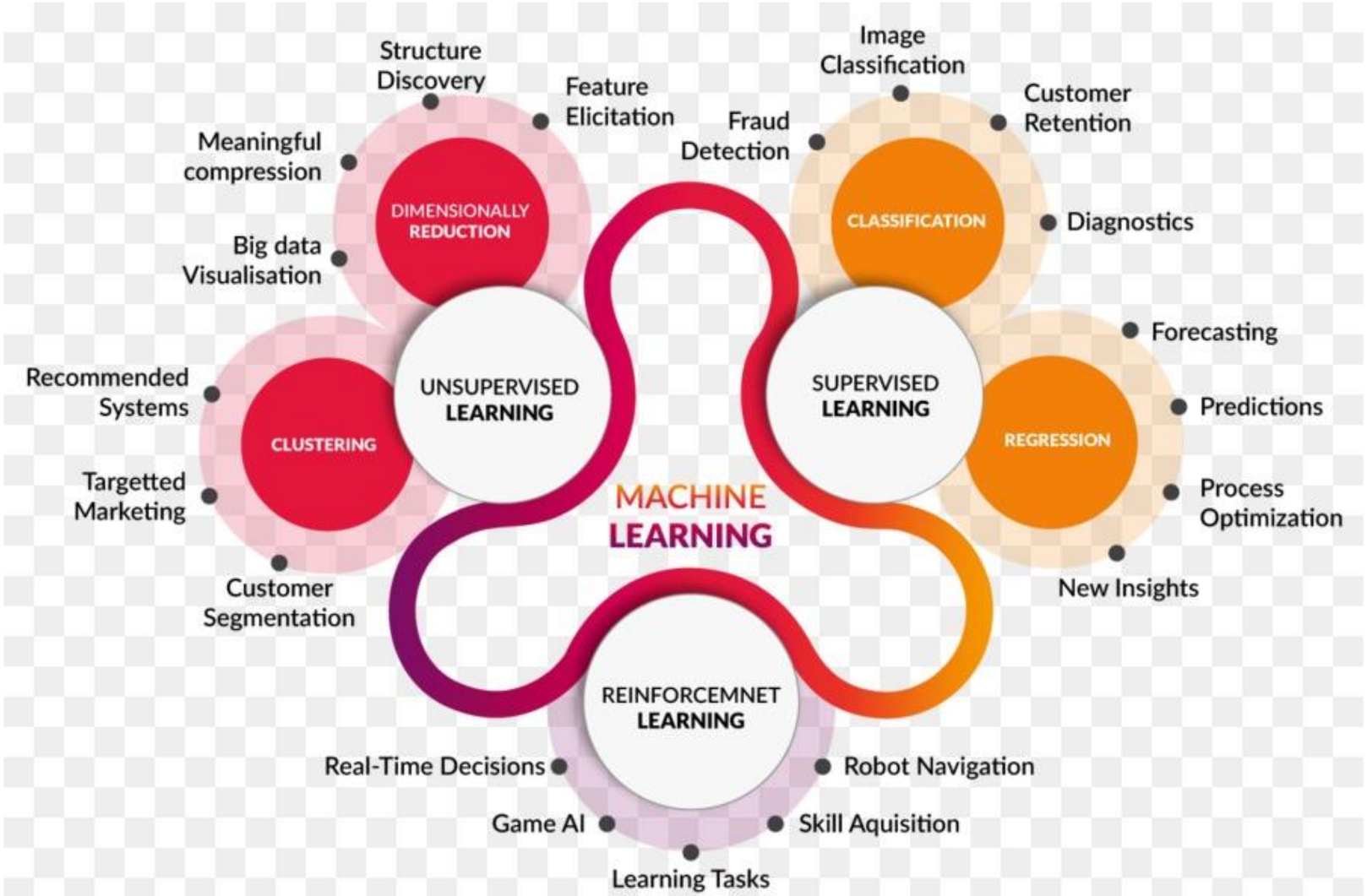
❖ الگوریتمهای بدون ناظر

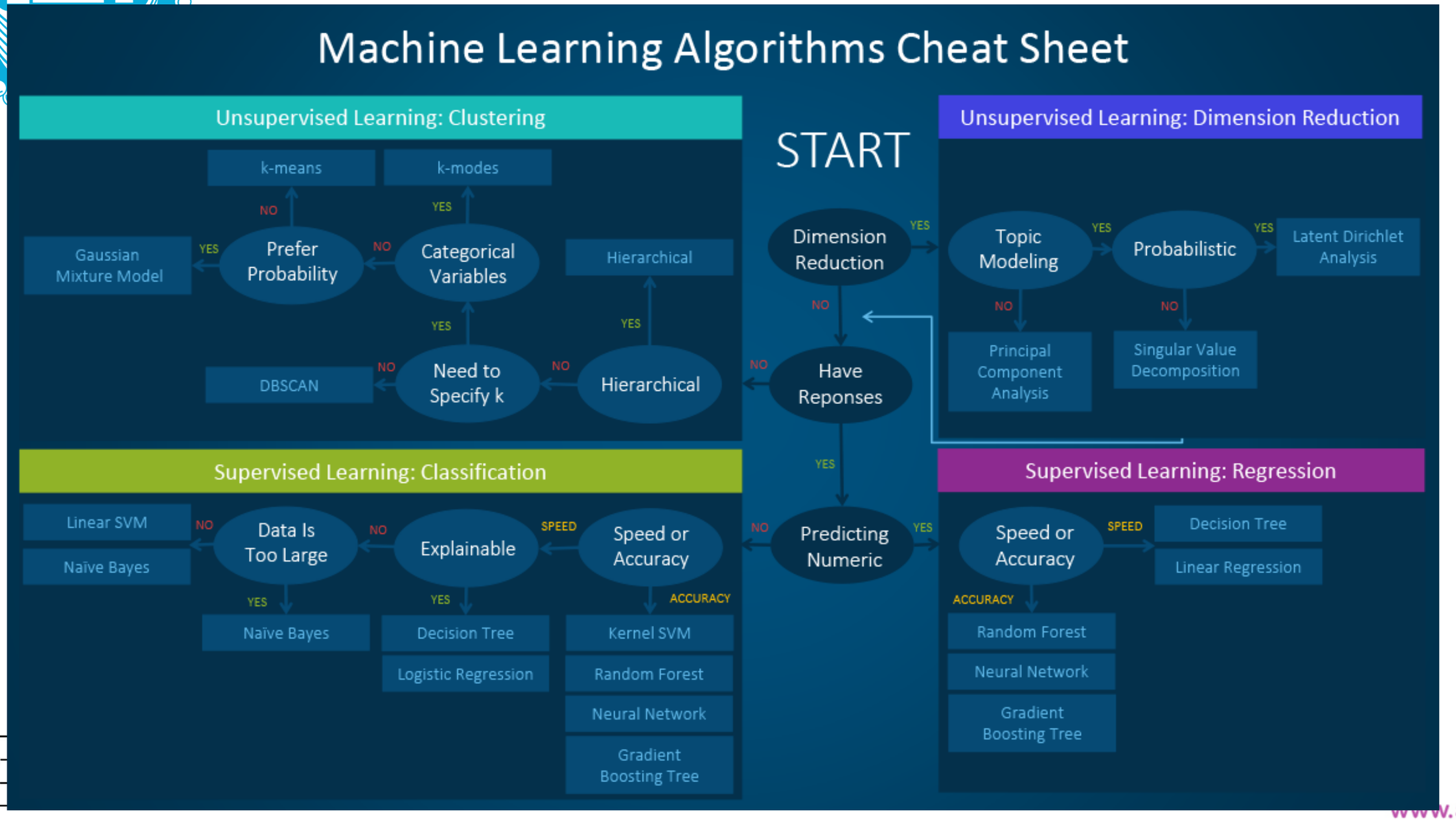


❖ الگوریتمهای یادگیری تقویتی

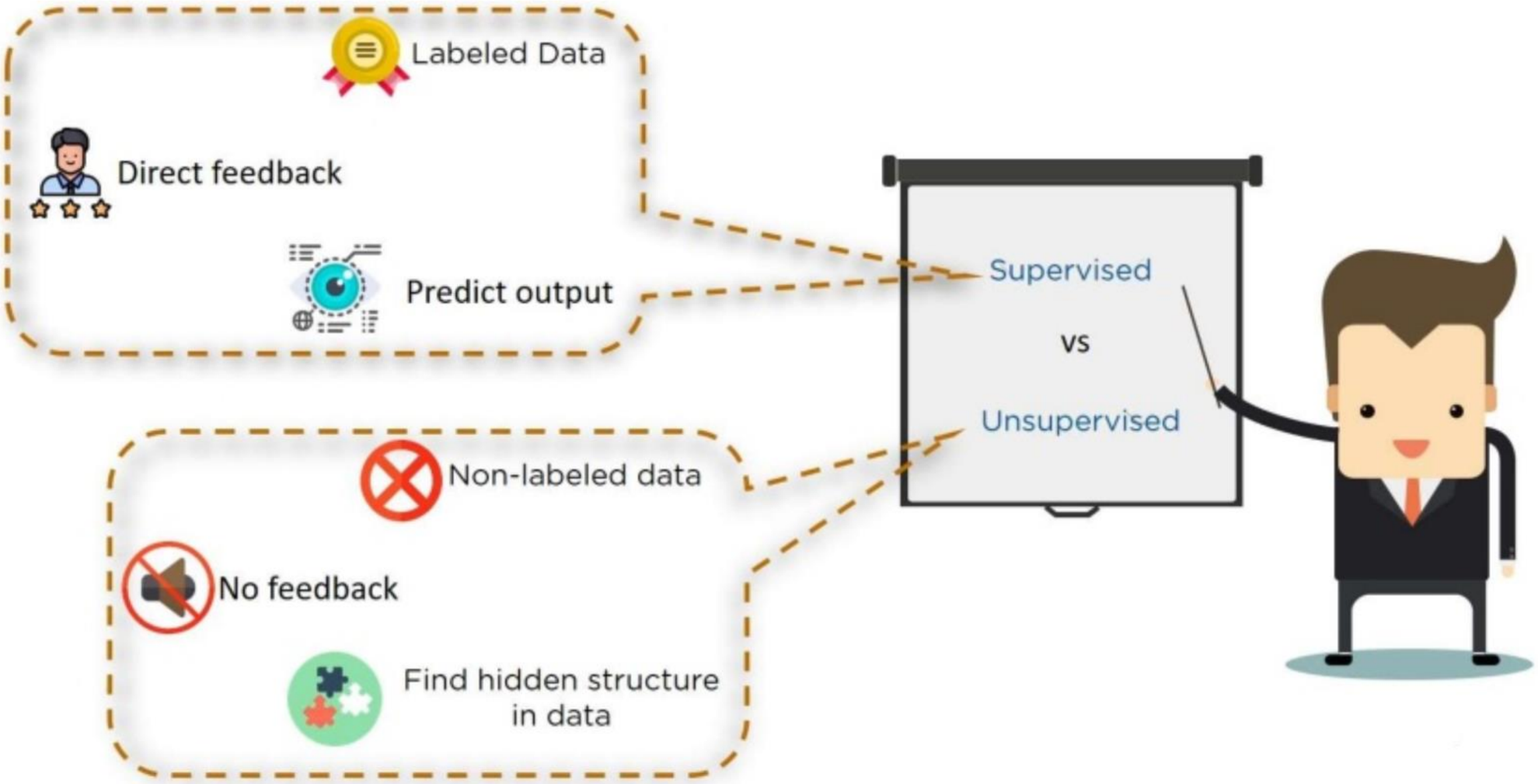


❖ چه الگوریتمهایی داریم؟




Machine Learning Algorithms Cheat Sheet
START
Unsupervised Learning: Clustering
k-means
k-modes
NO
YES
Gaussian Mixture Model
YES
Prefer Probability
NO
Categorical Variables
Hierarchical
YES
YES
DBSCAN
NO
Need to Specify k
NO
Hierarchical
Unsupervised Learning: Dimension Reduction
Dimension Reduction
YES
Topic Modeling
YES
Probabilistic
YES
Latent Dirichlet Analysis
NO
NO
NO
Principal Component Analysis
Singular Value Decomposition
NO
Have Reponses
YES
Supervised Learning: Classification
Linear SVM
NO
Data Is Too Large
NO
Explainable
SPEED
Speed or Accuracy
Naïve Bayes
YES
YES
ACCURACY
Naïve Bayes
Decision Tree
Kernel SVM
Logistic Regression
Random Forest
Neural Network
Gradient Boosting Tree
NO
Predicting Numeric
YES
Supervised Learning: Regression
Speed or Accuracy
SPEED
Decision Tree
Linear Regression
ACCURACY
Random Forest
Neural Network
Gradient Boosting Tree

❖تفاوت بین مدلهای با/بدون ناظر



❖ الگوریتم را چگونه انتخاب کنیم؟

Classification

Used when the output is categorical like 'YES' or 'NO'

Algorithms used

- Decision Tree
- Naïve Bayes
- Random Forest
- Logistic regression
- KNN

Clustering

Used when the data needs to be organized to find patterns in the case of 'product recommendation'

Algorithms used

- K Means

Regression

Used when a value needs to be predicted like the 'stock prices'

Algorithms used

- Linear Regression

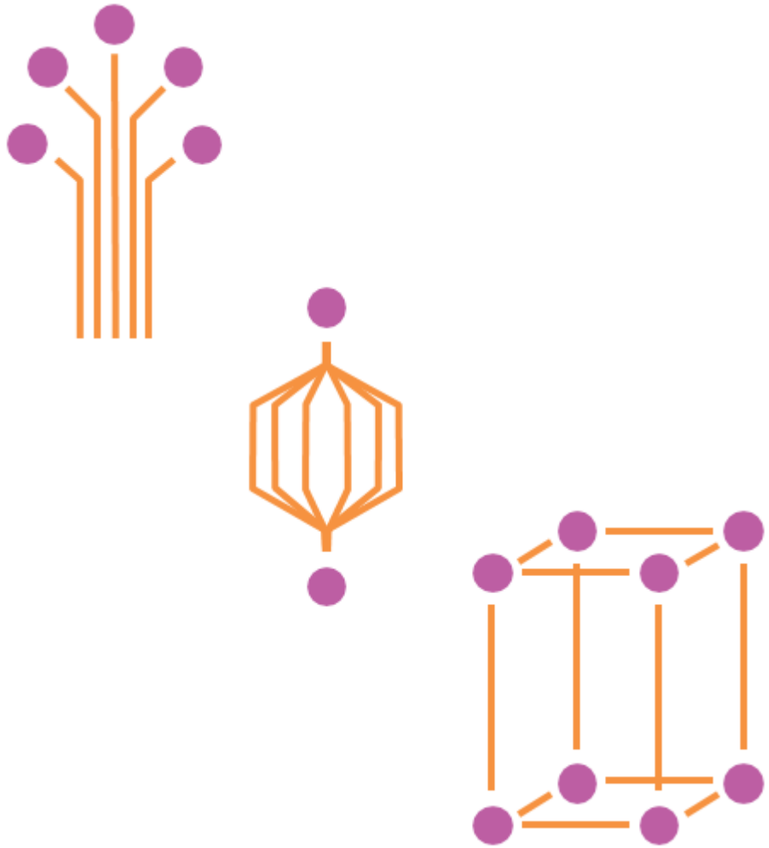
یادگیری

می گوییم یک برنامه کامپیوتری از تجربه E در مورد کارT یادگیری انجام داده است اگر عملکرد آن در صورت اندازه گیری با معیار P پس از این تجربه بهبود پیدا کند .

یادگیری بازی checkers	یادگیری تشخیص حروف دست نویس	یادگیری کنترل یک ربات
T :انجام بازی checkers P : درصد بازیهایی که بر حریف غلبه می کند T.E : انجام بازی بر علیه خودش	T : تشخیص حروف دست نویس در داخل تصویر P : درصد تشخیص صحیح حروف T.E : database ای ازکلمات دست نویس ودسته بندی هر کدام	T : هدایت یک ربات در بزرگراه چهار بانده با استفاده از دوربین P : میانگین فاصله طی شده قبل از اینکه خطایی رخ دهد T.E : با ضبط عملیات یک راننده و تصاویر پشت سر هم جاده بدست می آید

❖چند سوال در مورد یک مسئله یادگیری

- از چه الگوریتم یادگیری برای یک مسئله خاص میتوان استفاده کرد؟
- چه مقدار داده آموزشی موردنیاز است؟
- مثال آموزشی بعدی را چگونه باید انتخاب نمود؟
- چگونه میتوان مسئله یادگیری را به مسئله تقریب تابع تبدیل نمود؟
- آیا میتوان نحوه نمایش یادگیر را بطور خودکار تغییر داد تا با مسئله سازگاری بیشتری داشته باشد؟



بیگ دیتا و مفاهیم

Data & Analysis: Increasingly Complex!❖

scalability

DM

algorithms

ML

data volume too large — ***V**olume*

data rate too fast — ***V**elocity*

data too heterogeneous — ***V**ariability*

data too uncertain — ***V**eracity*

Data

Reporting
Ad-Hoc Queries
ETL/ELT

aggregation, selection
SQL, XQuery
MapReduce

MATLAB, R, Python Data Mining
Predictive/Prescriptive MATLAB, R, Python

Analysis

scalability

DM

algorithms

ML

Domain Expertise (e.g., Industry 4.0, Medicine, Physics, Engineering, Energy, Logistics)

Relational Algebra / SQL

Data Warehouse/OLAP

NF^2/XQuery

Resource Management

Hardware Adaptation

Fault Tolerance

Memory Management

Parallelization

Scalability

Memory Hierarchy

Data Analysis Language Compiler

Query Optimization

Data Flow

Real-Time

Control Flow

Indexing

Mathematical Programming

Linear Algebra

Stochastic Gradient Descent

Error Estimation

Active Sampling

Regression

Monte Carlo

Statistics

Sketches

Hashing

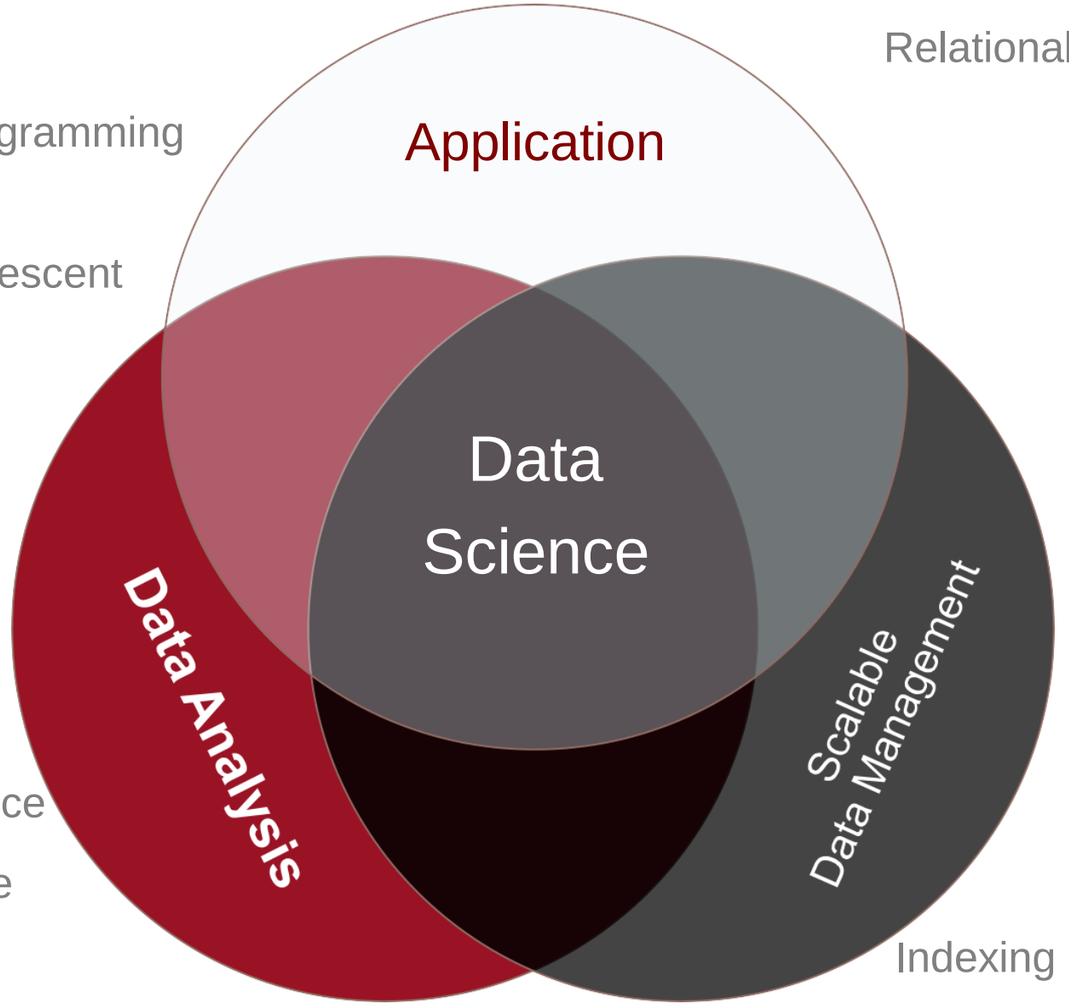
Convergence

Decoupling

Iterative

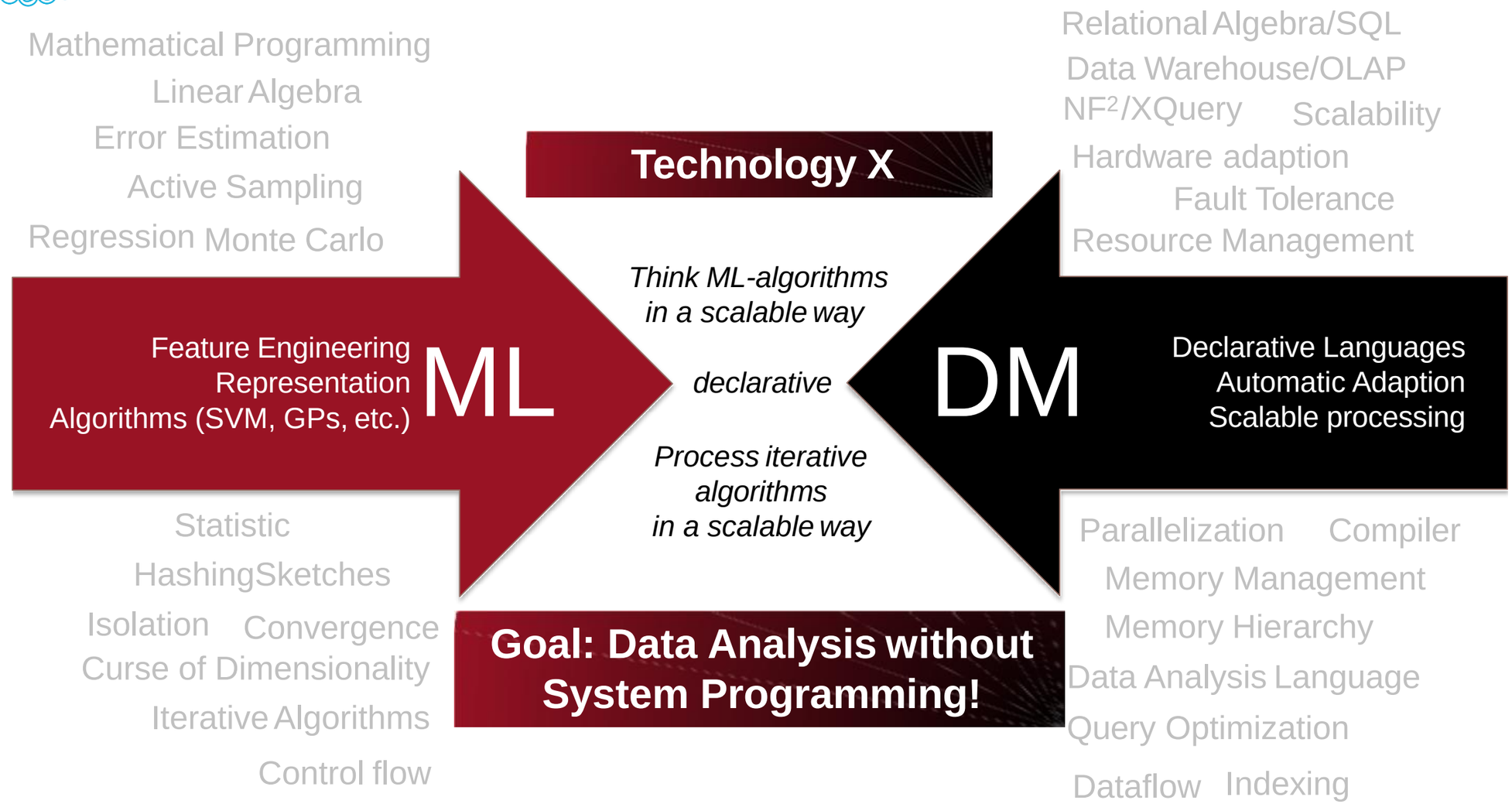
Algorithms

Curse of Dimensionality



New Technology to the Rescue!

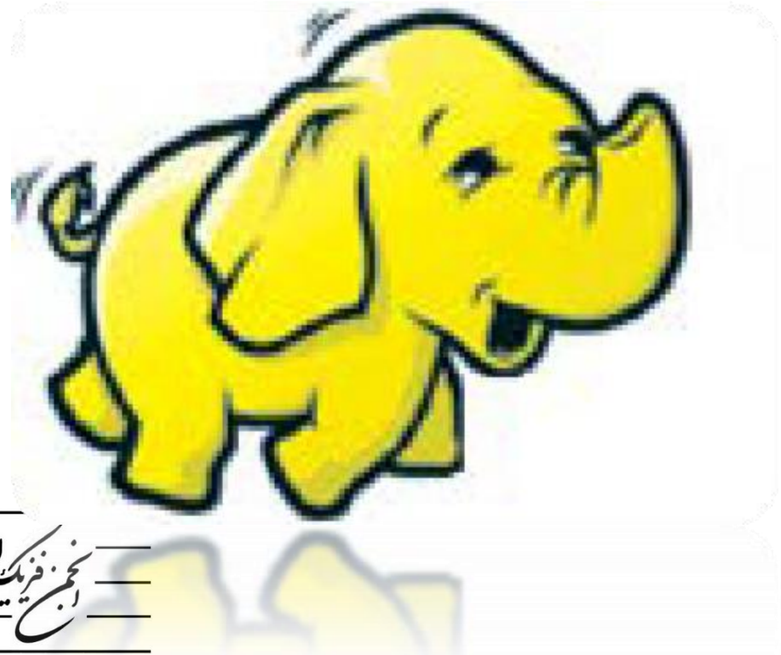
Machine Learning + Data Management = X❖



چقدر سریع؟

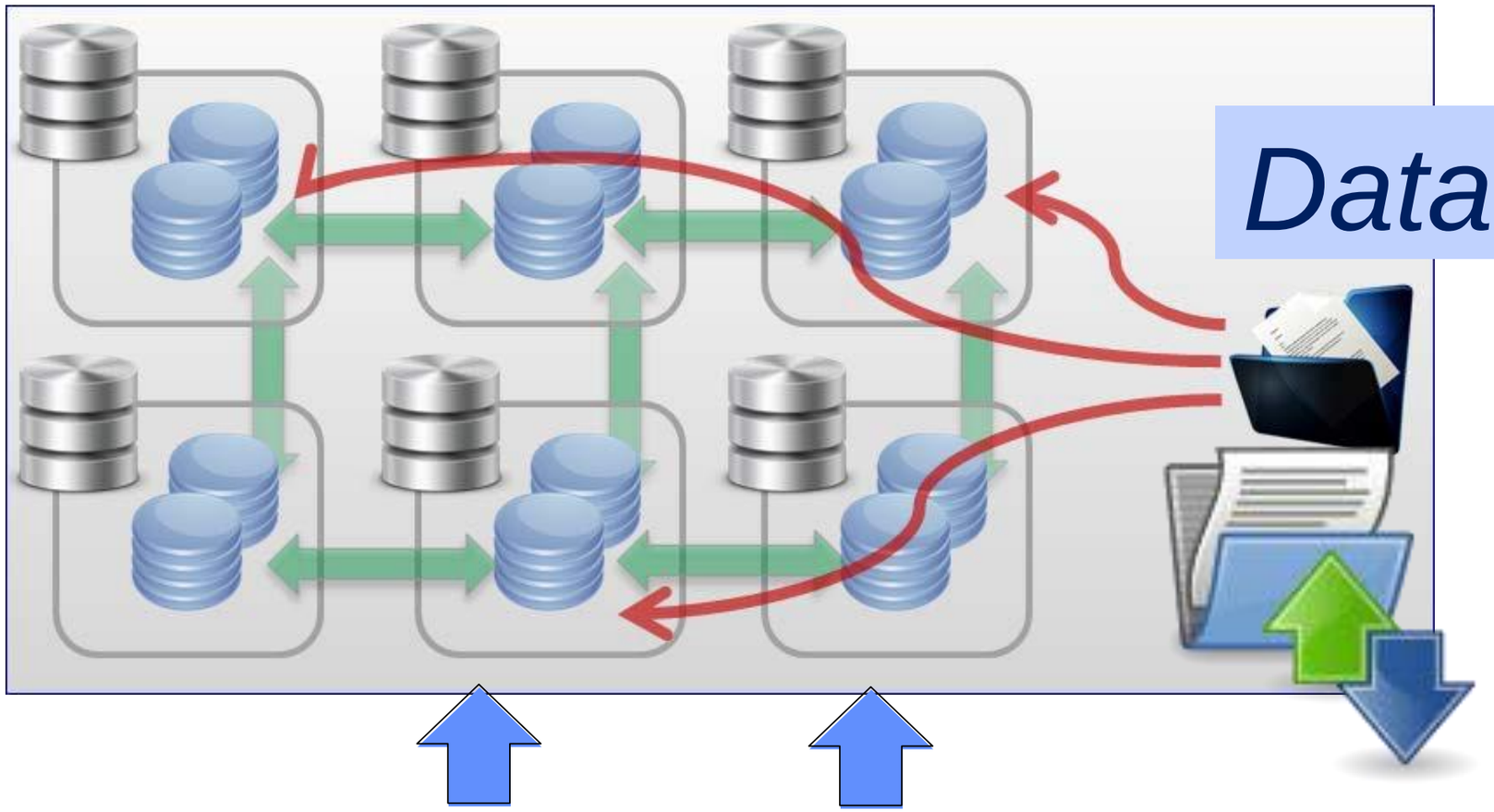
Big Data Stack❖

Apache Hadoop is an open source software framework for storage and large scale processing of data-sets on clusters of commodity hardware



Hadoop was created by Doug Cutting and Mike Cafarella in 2005, Named the project after son's toy elephant

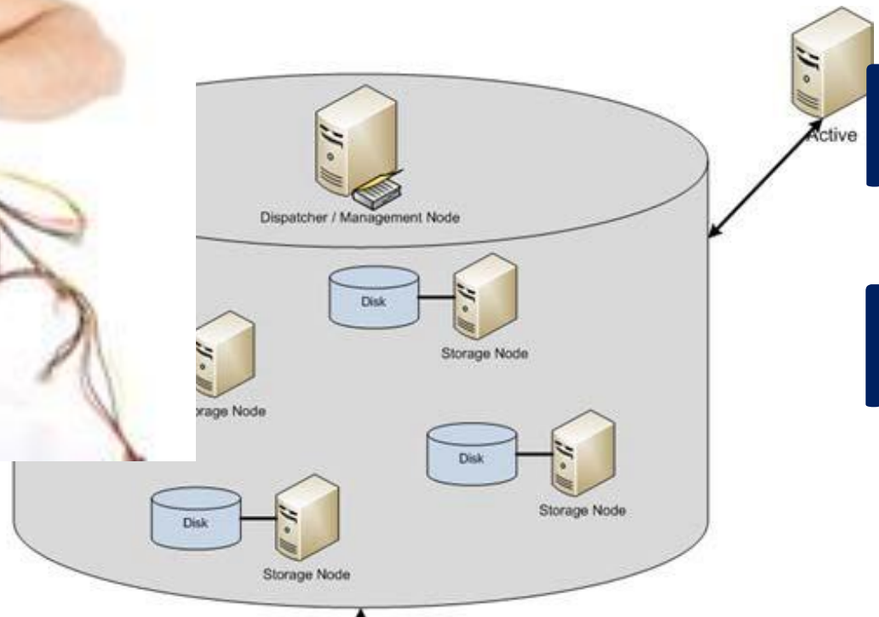
❖ انتقال محاسبات نزد داده



Scalability at Hadoop's core!❖

❖ قابلیت اعتماد

Reliability!
Reliability!
Reliability!

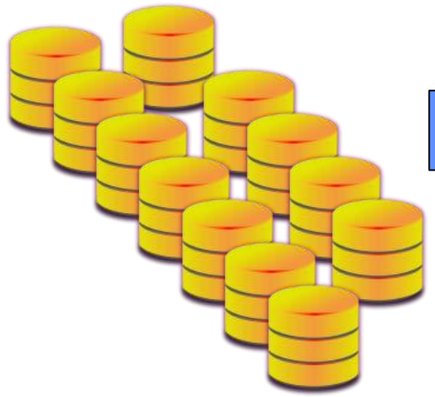


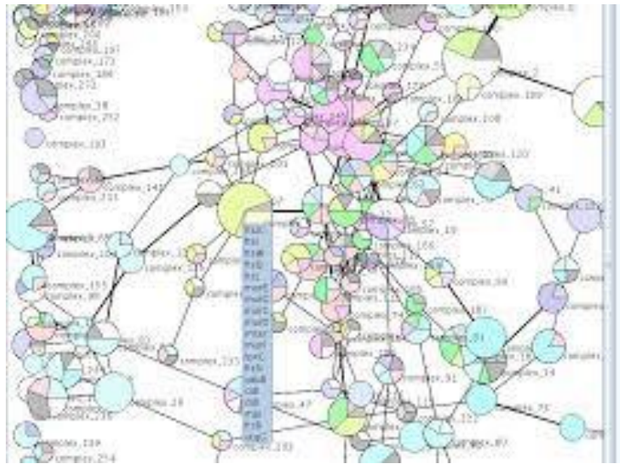
Hadoop Distributed File System(HDFS)

Once a year

365 Computers

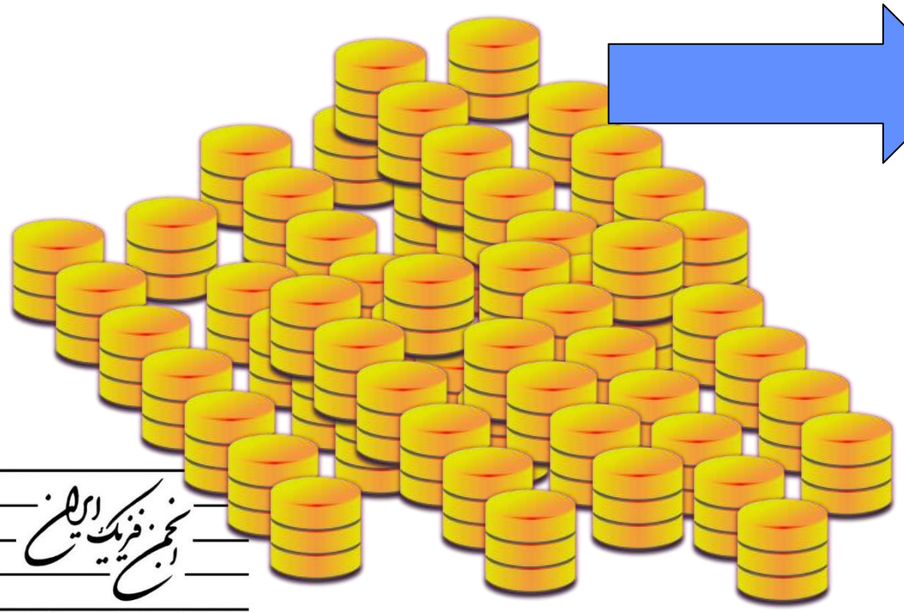
Once a day

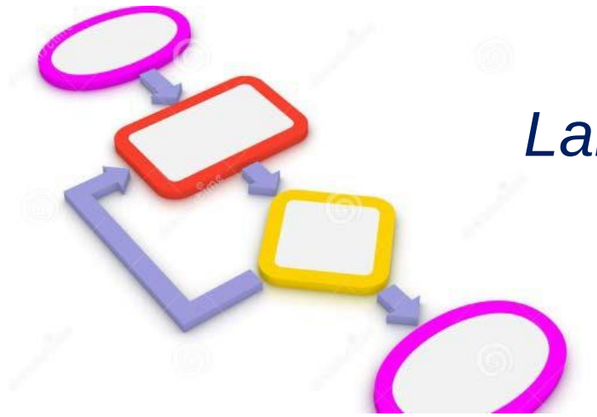




Small Data & Complex Algorithm

Vs.





Large Data & Simple Algorithm

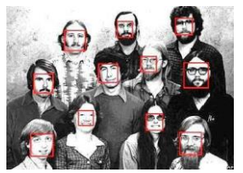
Data → Actionable Knowledge

That is roughly the problem that Machine Learning addresses!

Data → Knowledge

- Is this email spam or no spam?

- Is there a face in this picture?



- Should I lend money to this customer given his spending behaviour?

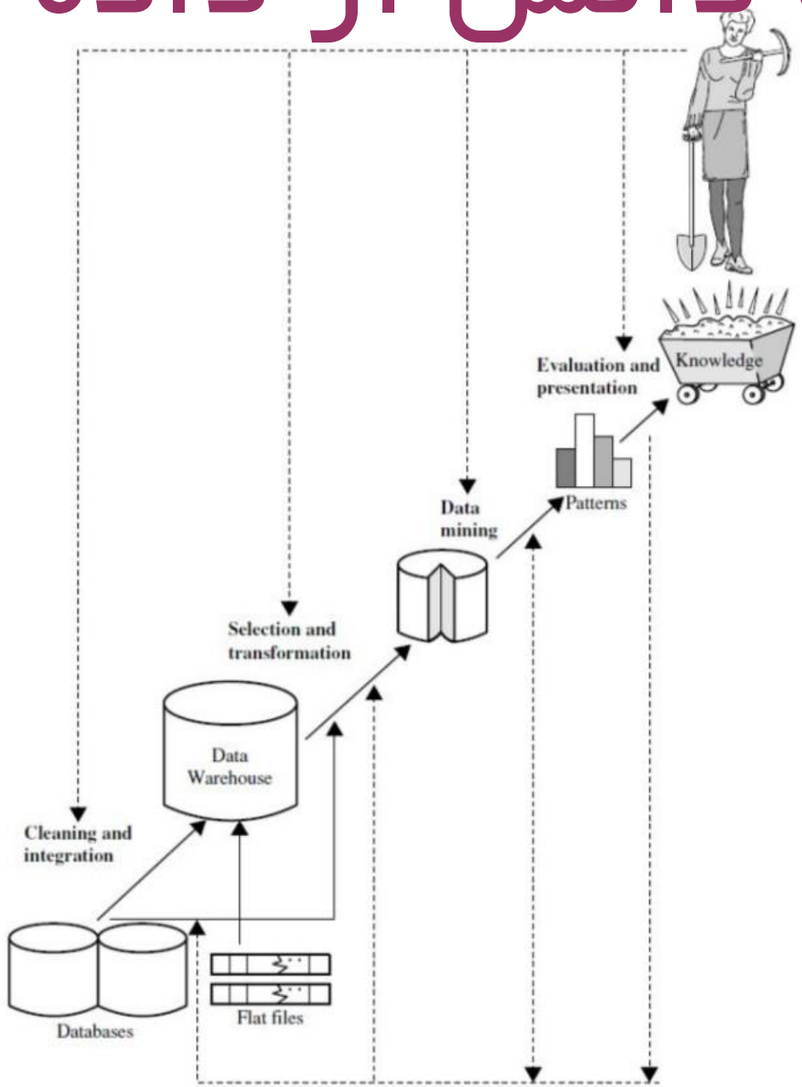
- Knowledge is not concrete

- Spam is an abstraction
- Face is an abstraction
- Who to lend to is an abstraction

You do not find spam, faces, and financial advice in datasets, you just find bits!

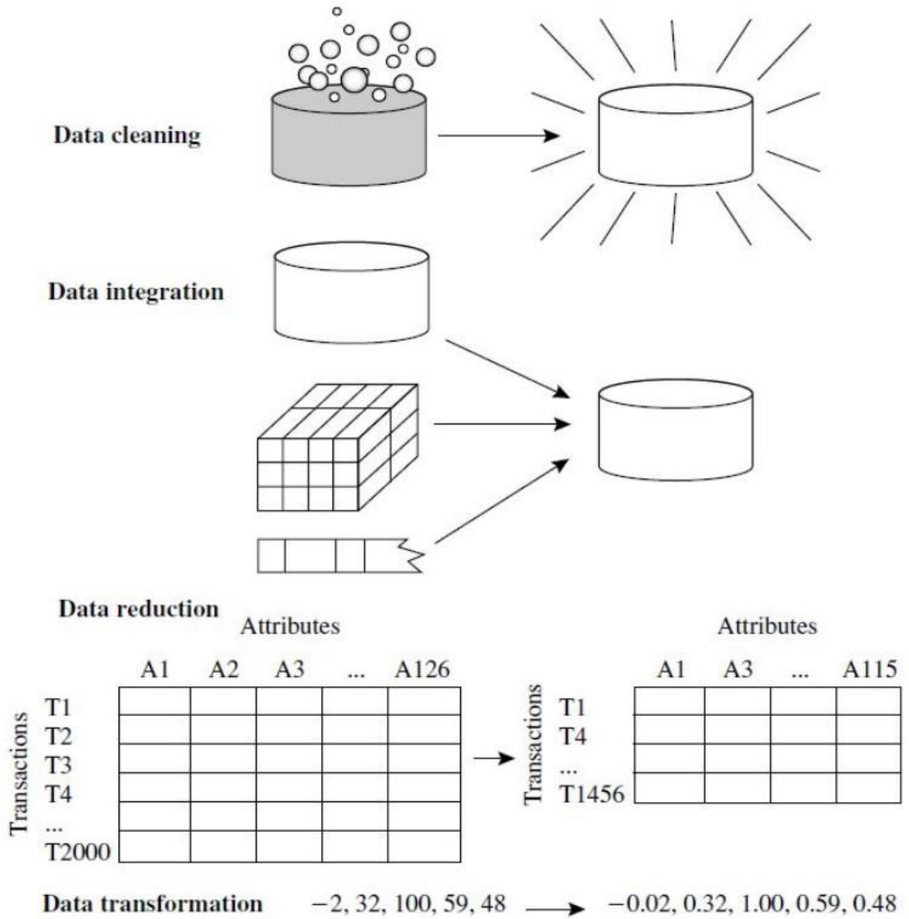
کشف دانش از داده ها

- Preprocessing
- Machine Learning
- Result validation



Preprocessing

- Data cleaning
- Data integration
- Data reduction, e.g., sampling
- Data transformation, e.g., normalization



Functionality & Validation

- Classification and regression (supervised learning)
- Clustering (unsupervised learning)
- Mining the frequent patterns
- Outlier detection

- Needs to evaluate the performance of the model on some criteria.
- Depends on the application and its requirements.

What is Spark?❖

- What is Spark ?

  High level Architecture
  How does it Work ? RDD and Operations Hadoop MapReduce
  DAG (Directed Acyclic Graph) Run Mode of Spark Programming model

- Machine Learning With Spark

  MLLib Library
  Types of Machine Learning ML Architecture
  Comparison with other tools

- Other Applications

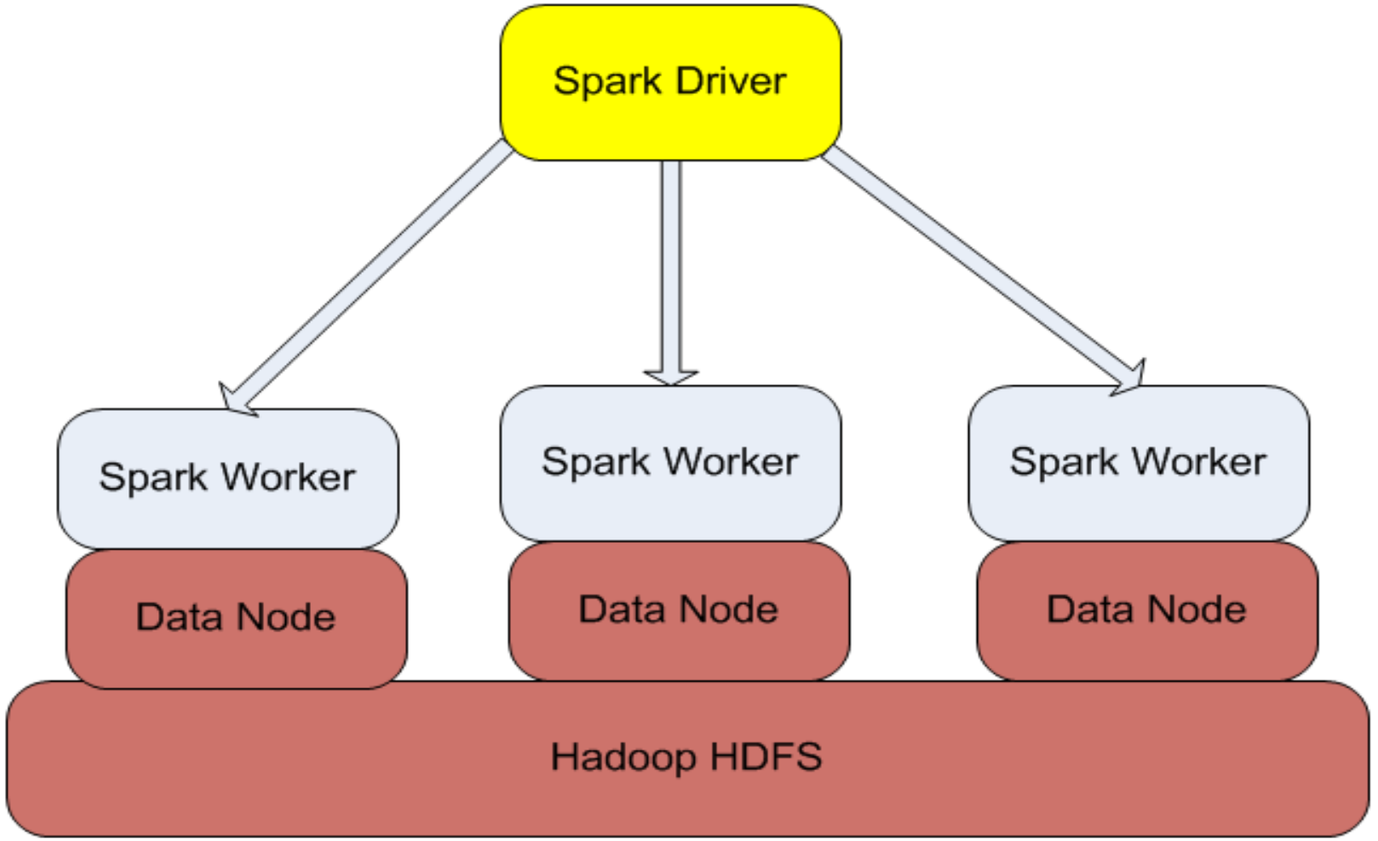
What is Spark?❖

- ***Definition***

"Apache Spark is an open source big data processing framework built around speed, ease of use, and sophisticated analytics. It was originally developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010 as an Apache project."

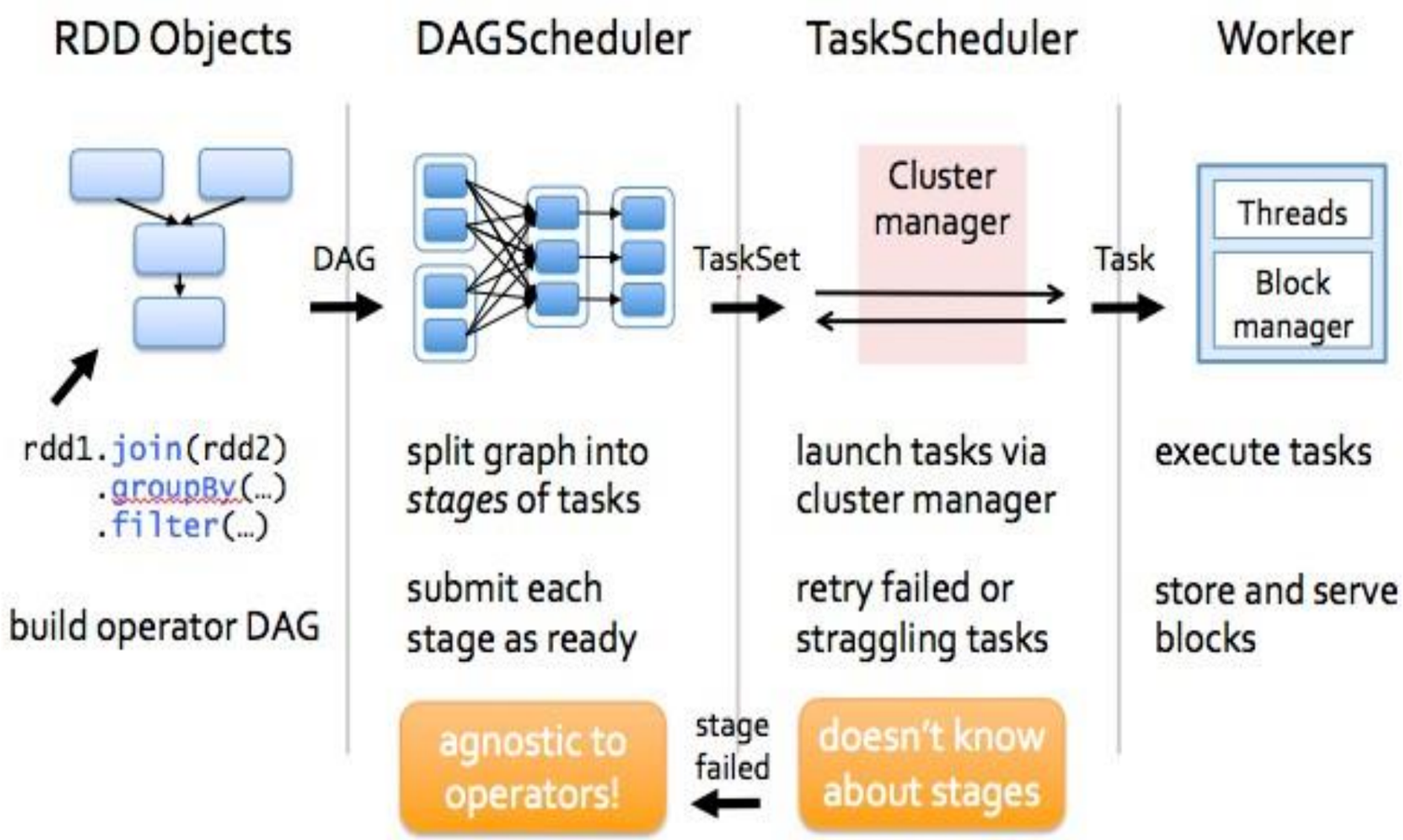
Source : http://www.infoq.com/articles/apache-spark-introduction

معماری Spark



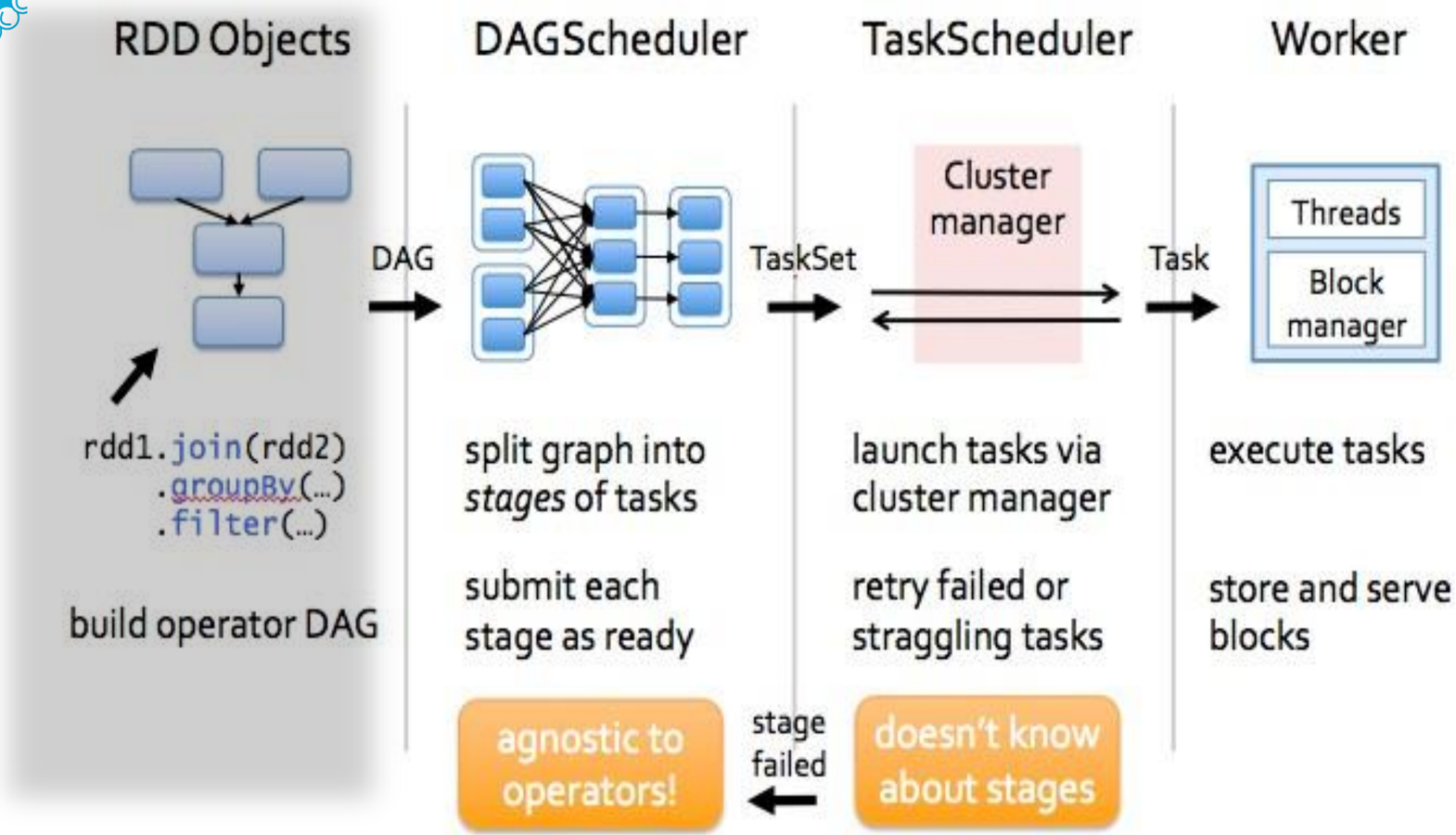
Source : https://bighadoop.wordpress.com/2014/04/03/apache-spark-a-fast-big-data-analytics-engine/

❖Spark چگونه کار می کند؟



Source: https://www.sigmoid.com/apache-spark-internals/

RDD and Operations



Source : http://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html

❖RDD و عملیاتهای آن

- *Definition*

"A Resilient Distributed Dataset (RDD), the basic abstraction in Spark. Represents an immutable, partitioned collection of elements that can be operated on in parallel. This class contains the basic operations available on all RDD"

- *Creating RDD:*
  - ✓ *From existing collection*

```
val collection = List("a", "b", "c", "d")
val rddFromCollection = sc.parallelize (collection)
```

  - ✓ *From Hadoop-based input sources*

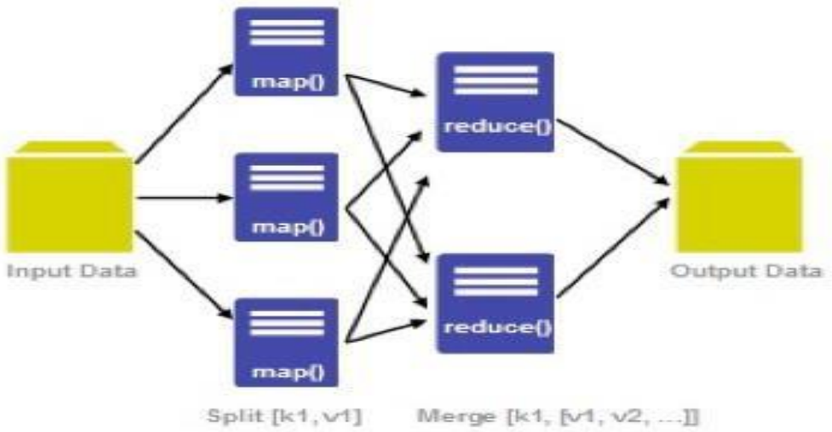
```
val rddFromTextFile= sc.textFile ("List ")
```

Source : http://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html

❖مدل برنامه نویسی Map/Reduce

- ***Definition***

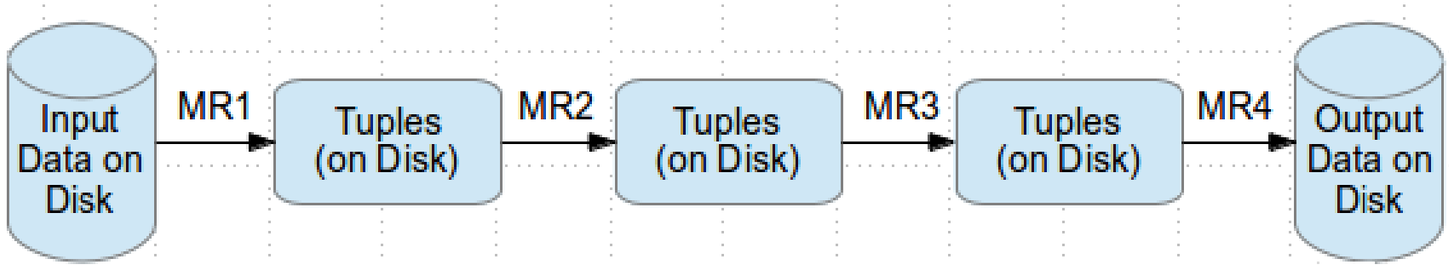
"The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job … and the second is the reduce."



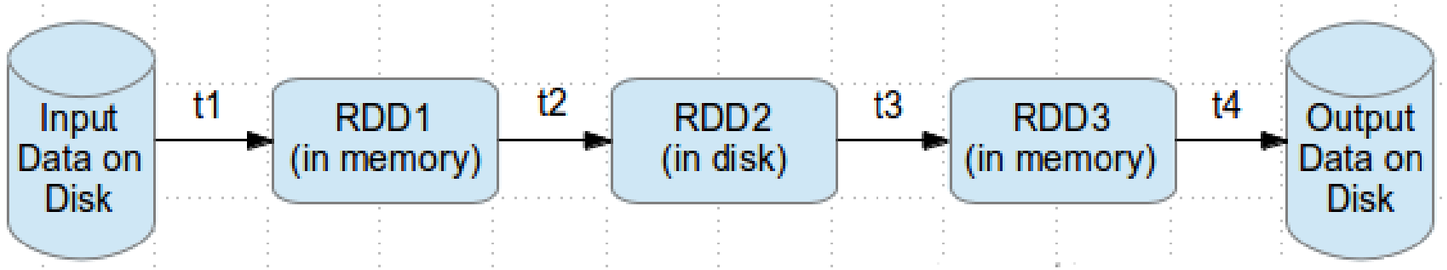
Source: https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/

❖RDD و عملیاتهای آن

Case of MapReduce :



Case of RDD:



Source: http://www.thecloudavenue.com/

Spark Machine Learning ❖

- ***MLLib Library :***

"MLlib is Spark's scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization Primitives"

Source: https://spark.apache.org

MLLib چیست؟

Algorithms:

- **classification:** logistic regression, linear support vector machine (SVM), naive Bayes
- **regression:** generalized linear regression (GLM)
- **collaborative filtering:** alternating least squares (ALS)
- **clustering:** k-means
- **decomposition:** singular value decomposition (SVD), principal component analysis (PCA)

Why MLlib?

Scikit-Learn

Algorithms:

- **classification:** SVM, nearest neighbors, random forest, …
- **regression:** support vector regression (SVR), ridge regression, Lasso, logistic regression, …
- **clustering:** k-means, spectral clustering, …
- **decomposition:** PCA, non-negative matrix factorization (NMF), independent component analysis (ICA), …

Apache Mahout

Algorithms:

- **classification:** logistic regression, naive Bayes, random forest, …
- **collaborative filtering:** ALS, …
- **clustering:** k-means, fuzzy k-means, …
- **decomposition:** SVD, randomized SVD, …

LIBLINEAR?

Mahout?

H2O?

Vowpal Wabbit?

MATLAB?

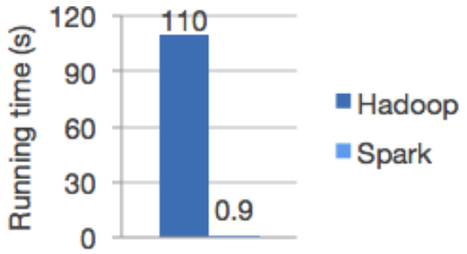
R?

scikit-learn?

Weka?

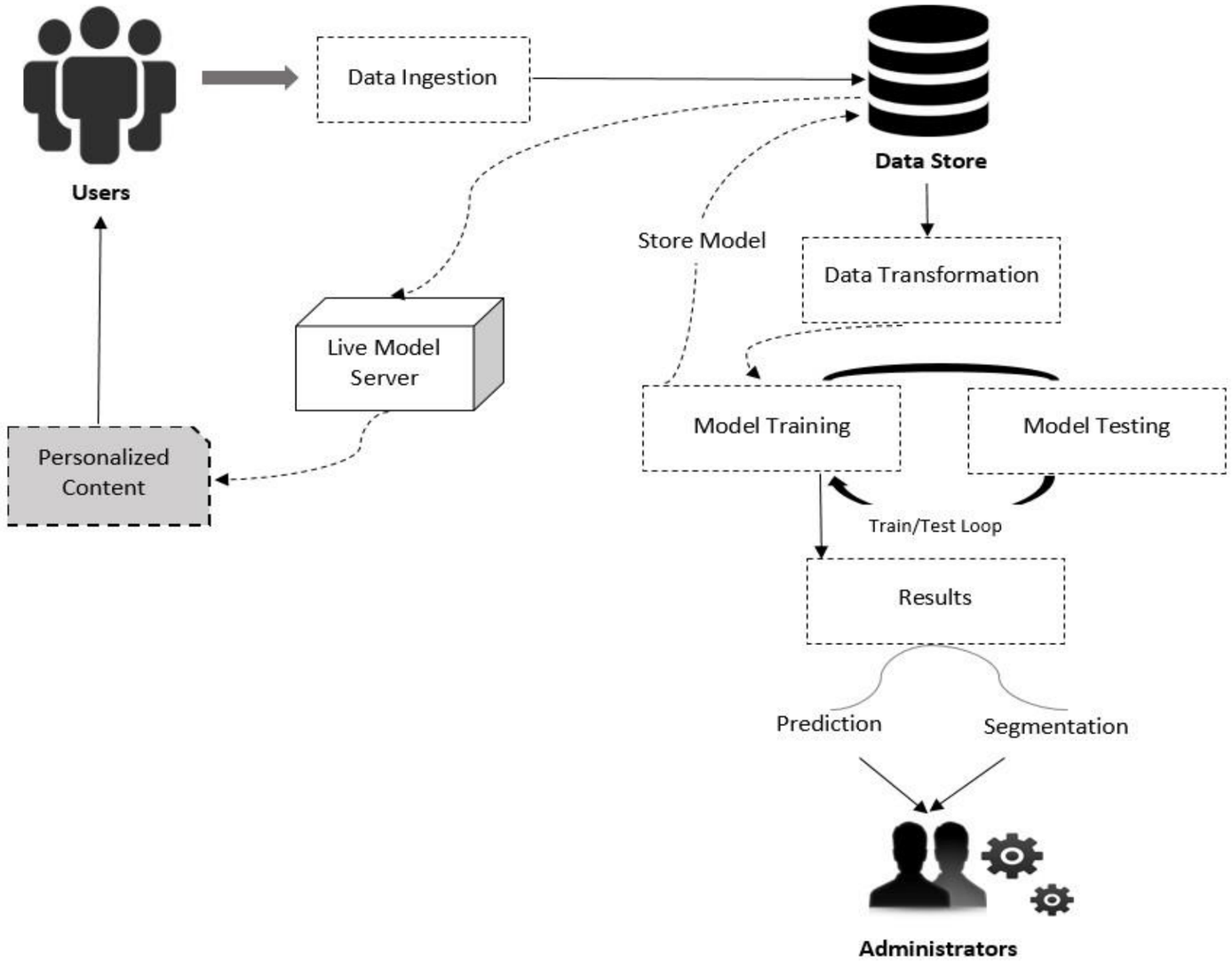
Why MLlib

- It is built on Apache Spark, a fast and general engine for large-scale data processing.
  - Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

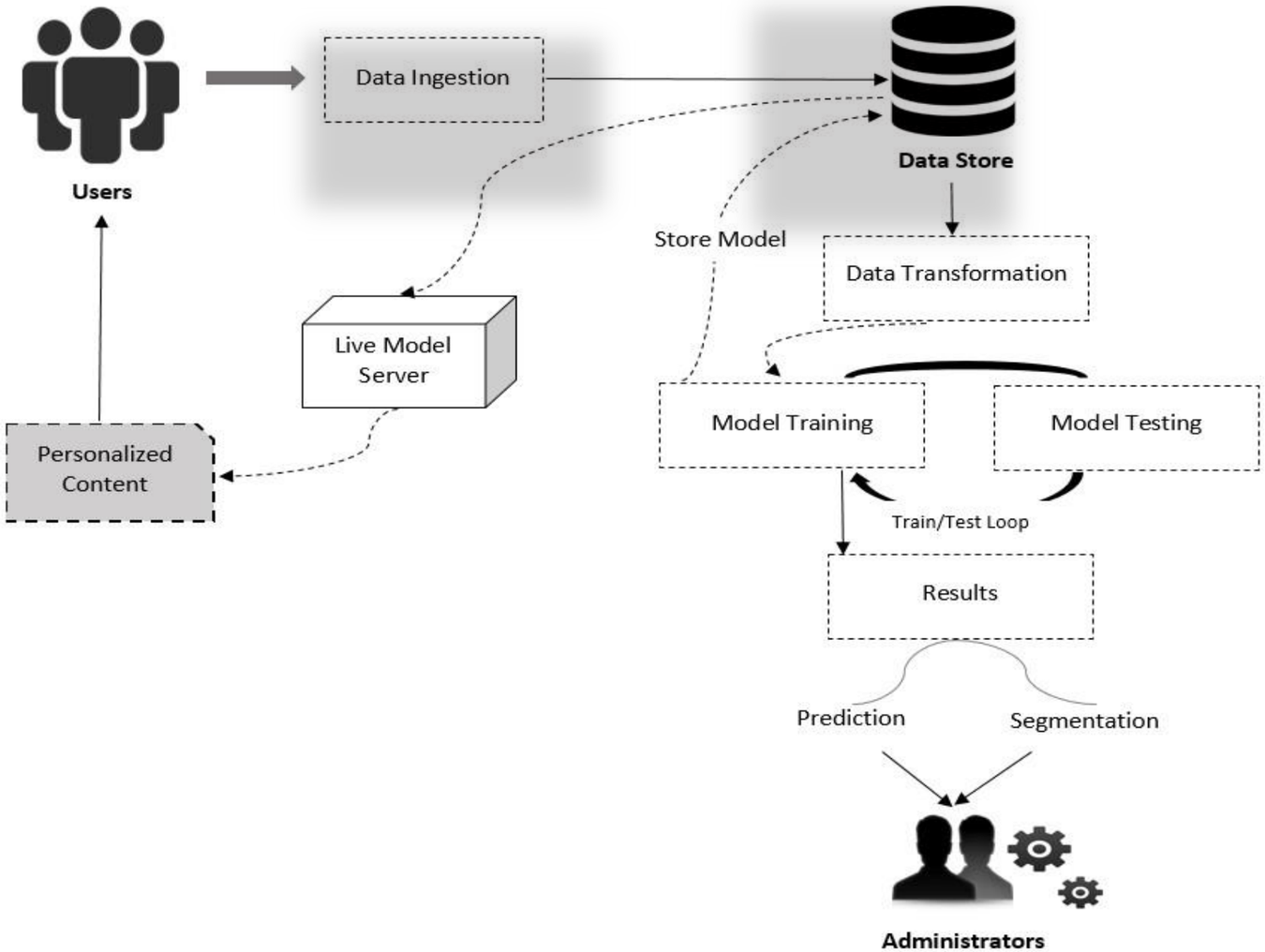


  - Write applications quickly in Java, Scala, or Python.

معماری Mllib



Data Ingestion and Storage



Data Ingestion and Storage ❖

- ***Data Ingestion:***

  - Browser, and mobile application event logs or accessing external web APIs

- ***Data Storage:***

  - HDFS, Amazon S3, and other filesystems; SQL databases such as MySQL or PostgreSQL; distributed NoSQL data stores such as HBase, Cassandra, and DynamoDB, …

Implementation of K-Means❖

Initialization:

- random
- k-means++
- k-means||

K-Means(Scala)❖

```scala
// Load and parse the data.
val data = sc.textFile("kmeans_data.txt")
val parsedData = data.map(_.split(' ').map(_.toDouble)).cache()

// Cluster the data into two classes using KMeans.
val clusters = KMeans.train(parsedData, 2, numIterations = 20)

// Compute the sum of squared errors.
val cost = clusters.computeCost(parsedData)
println("Sum of squared errors = " + cost)
```

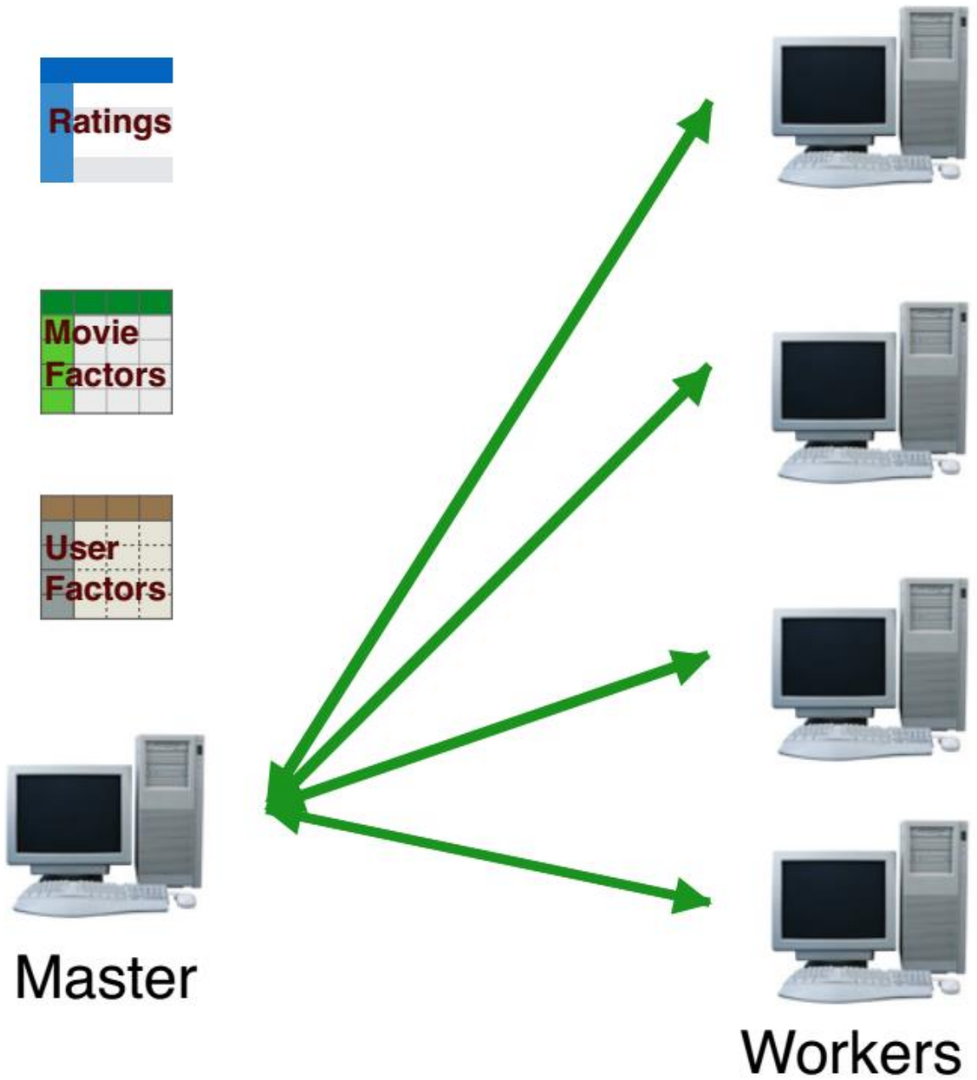
K-Means(Python)❖

```
# Load and parse the data
data = sc.textFile("kmeans_data.txt")
parsedData = data.map(lambda line:
                array([float(x) for x in line.split(' ‘)])).cache()

# Build the model (cluster the data)
clusters = KMeans.train(parsedData, 2, maxIterations = 10,
              runs = 1, initialization_mode = "kmeans||")

# Evaluate clustering by computing the sum of squared errors
def error(point):
    center = clusters.centers[clusters.predict(point)]
    return sqrt(sum([x**2 for x in (point - center)]))

cost = parsedData.map(lambda point: error(point))
          .reduce(lambda x, y: x + y)
print("Sum of squared error = " + str(cost))
```

Dimension Reduction(K-Means++)❖

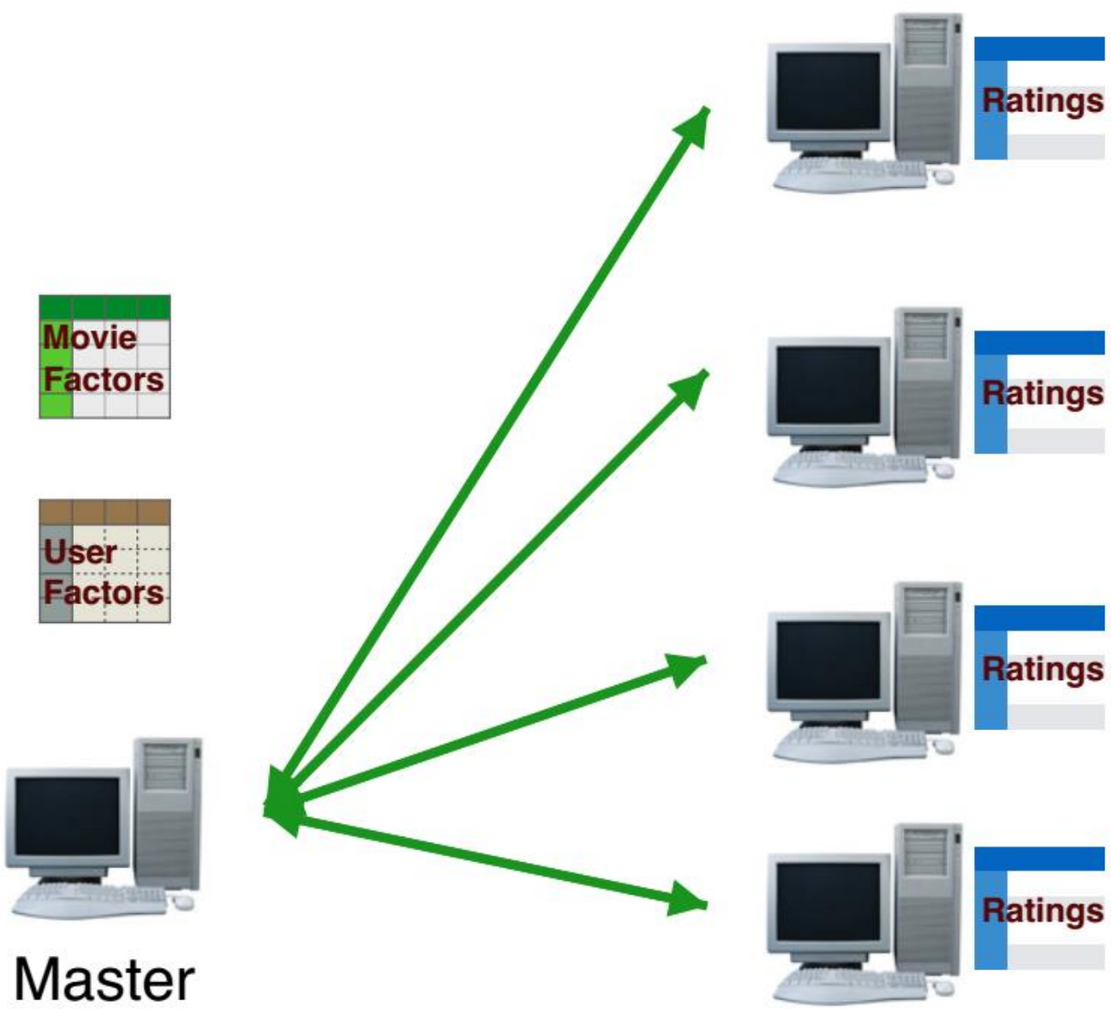
```
// compute principal components
val points: RDD[Vector] = ...
val mat = RowRDDMatrix(points)
val pc = mat.computePrincipalComponents(20)

// project points to a low-dimensional space
val projected = mat.multiply(pc).rows

// train a k-means model on the projected data
val model = KMeans.train(projected, 10)
```

Broadcast Everything❖



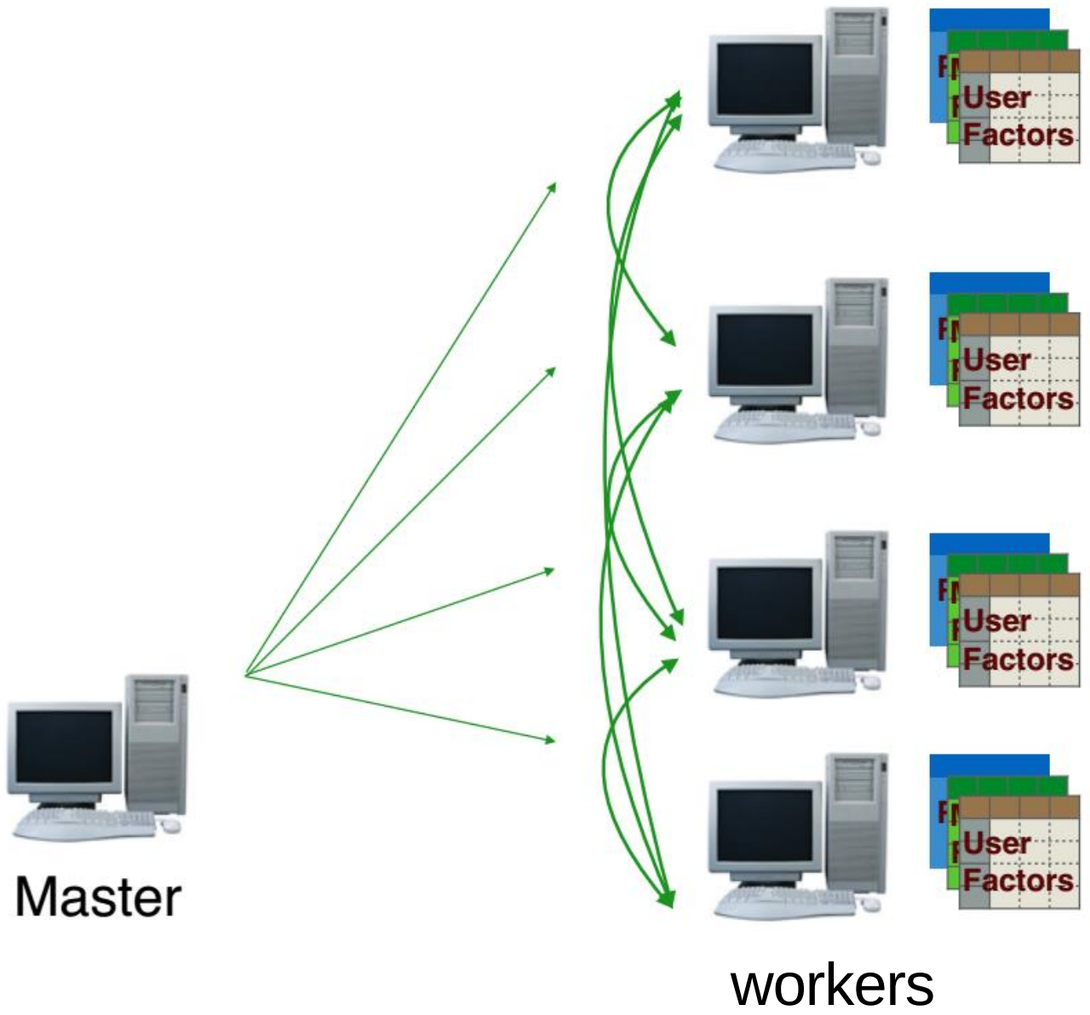
- Master loads (small) data file and initializes models.
- Master broadcasts data and initial models.
- At each iteration, updated models are broadcast again.
- Works OK for small data.
- Lots of communication overhead - doesn't scale well.

Data Parallel

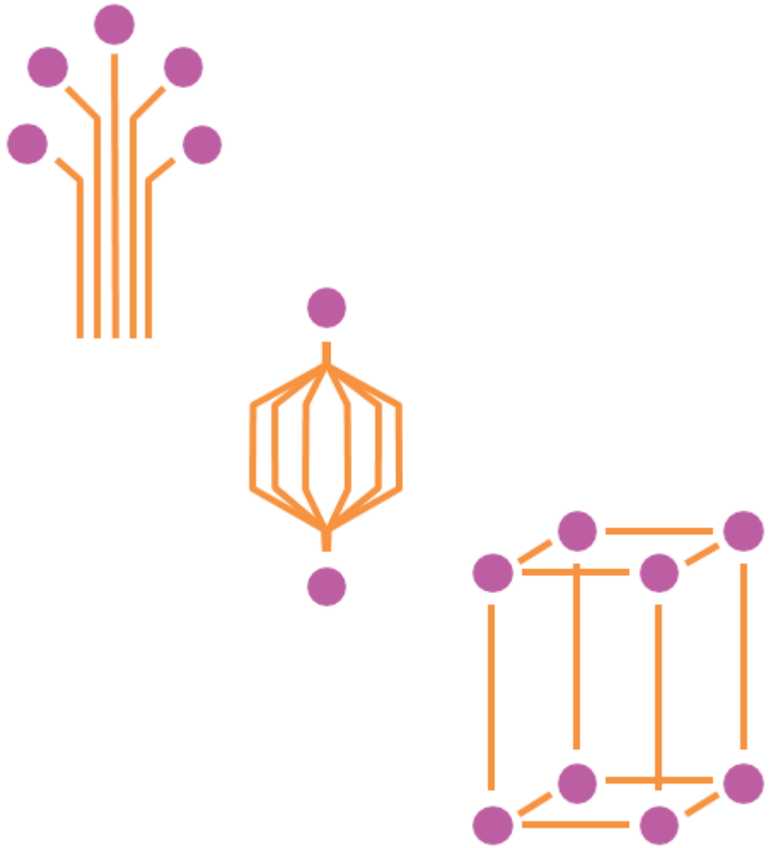


- Workers load data
- Master broadcasts initial models
- At each iteration, updated models are broadcast again
- Much better scaling
- Works on large datasets
- Works well for smaller models. (low K)

Fully Parallel❖

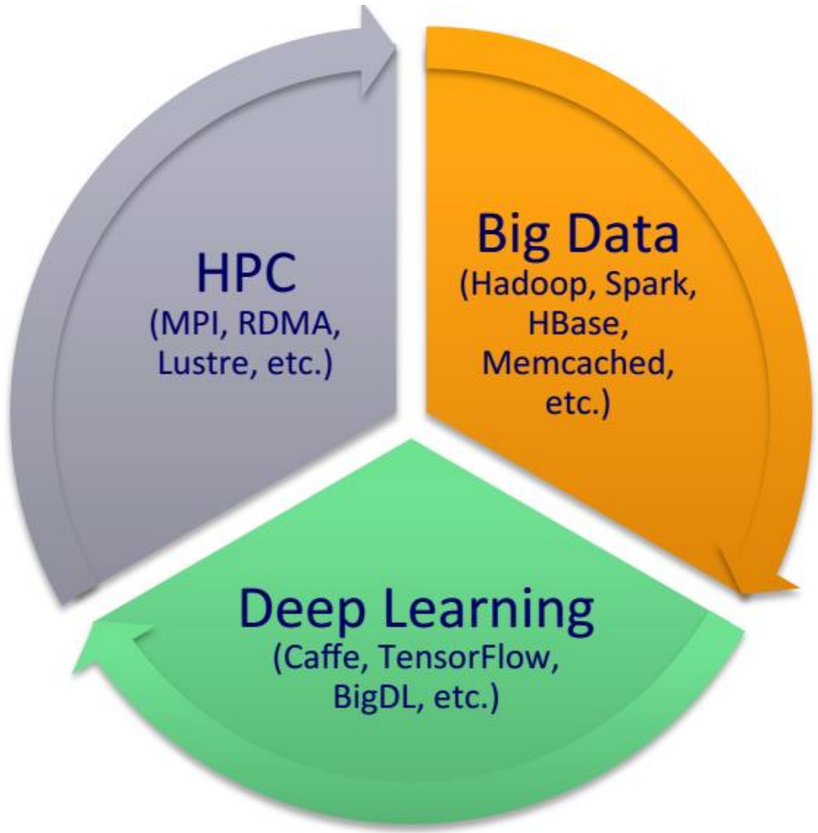


- Workers load data
- Models are instantiated at workers.
- At each iteration, models are shared via join between workers.
- Much better scalability.
- Works on large datasets



روشهای افزایش سرعت

❖رشد استفاده از HPC, Big Data, Deep Leraning



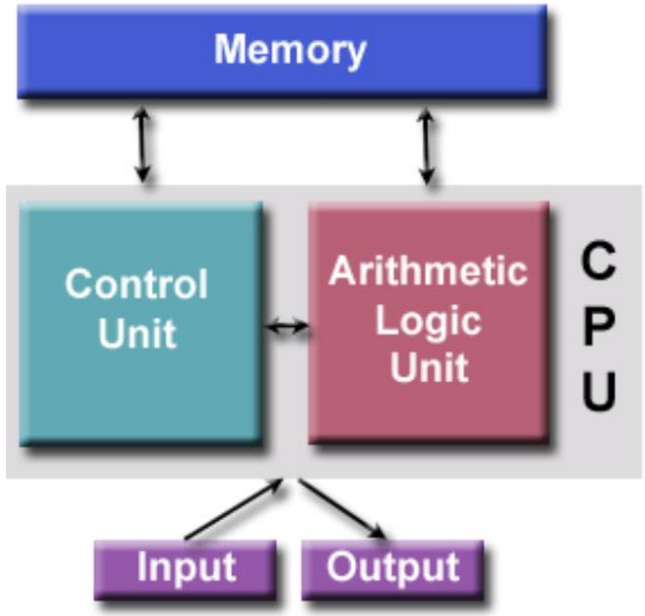
همگرایی بین HPC, Big Data, Deep Learning

❖ معماری von Neumann

ریاضیدان نخبه مجارستانی که پس از مقاله سال ۱۹۴۵ خود با موضوع «نیازمندیهای عمومی برای یک کامپیوتر عمومی» به شهرت رسید.

در مدل او برخلاف کامپیوتر های قدیمی که برنامه نویسی با "Hard Wiring" انجام می شد، هر دوی دستورالعملهای برنامه و داده های مورد نیاز آن در حافظه الکترونیکی نگهداری می شوند.

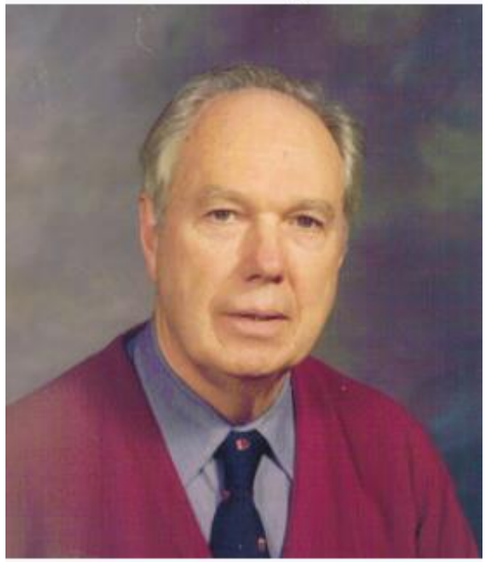
John von Neumann circa 1940s
(Source: LANL archives)



❖ دسته بندی Flynn

معروف ترین دسته بندی موجود برای انواع برنامه های کامپیوتری در سال ۱۹۶۶ توسط آقای مایکل فلین ارائه شد.

Michael Flynn

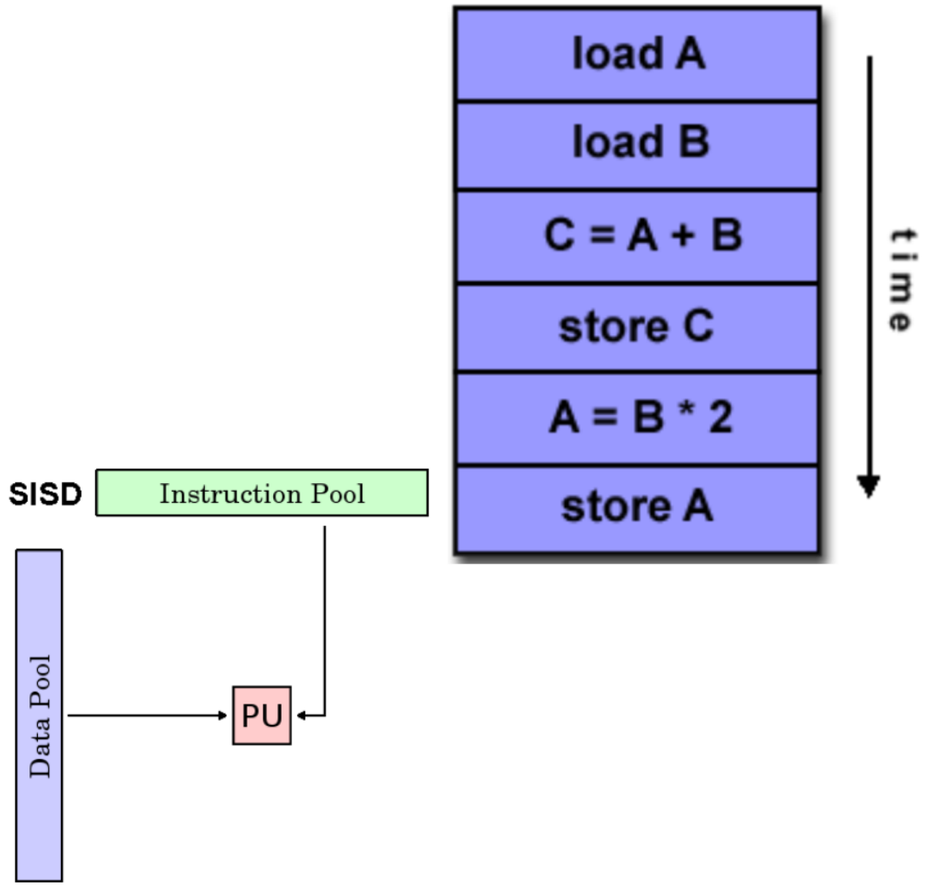


May 20, 1934 (age 84)

S I S D Single Instruction stream Single Data stream	**S I M D** Single Instruction stream Multiple Data stream
M I S D Multiple Instruction stream Single Data stream	**M I M D** Multiple Instruction stream Multiple Data stream

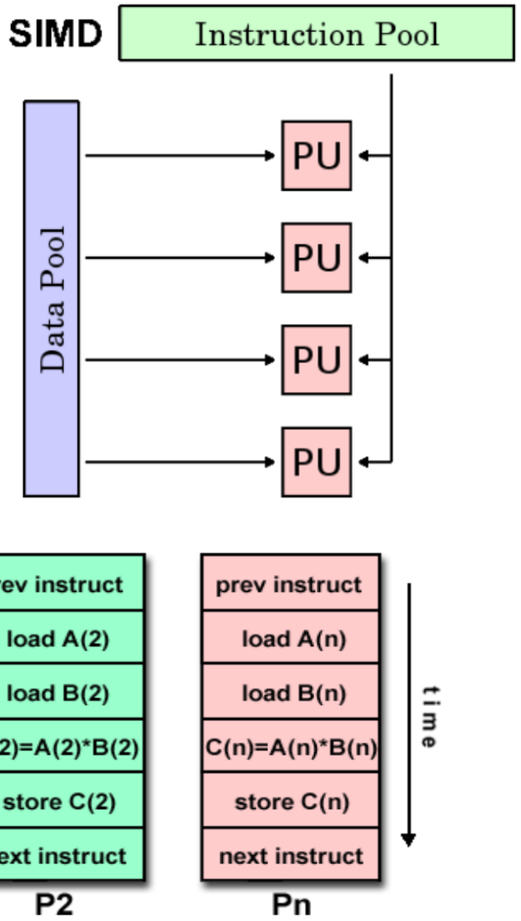
Single Instruction, Single Data (SISD)

- A serial (non-parallel) computer
- **Single Instruction**: Only one instruction stream is being acted on by the CPU during any one clock cycle
- **Single Data**: Only one data stream is being used as input during any one clock cycle
- Deterministic execution
- This is the oldest type of computer
- Examples: older generation mainframes, minicomputers, workstations and single processor/core PCs.



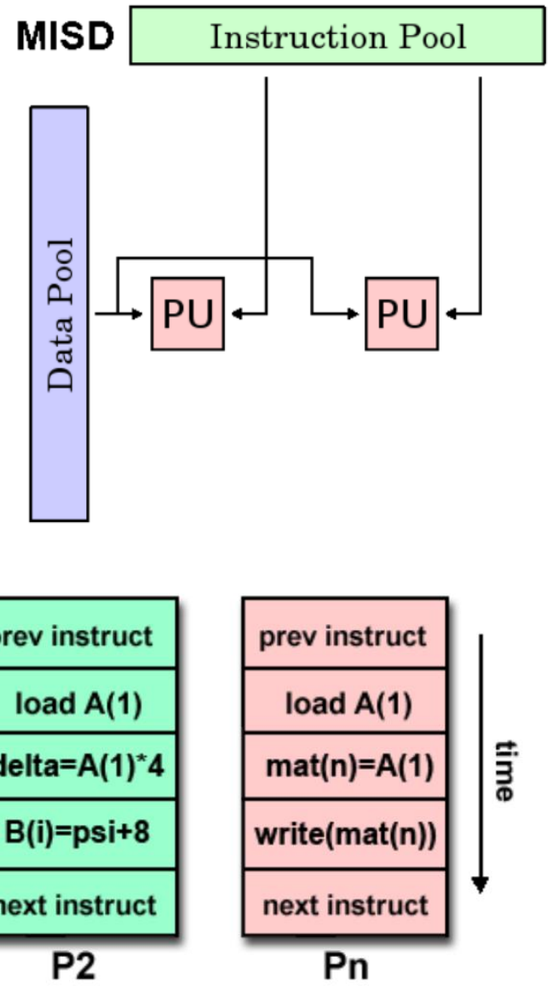
Single Instruction, Multiple Data (SIMD) ❖

- A type of parallel computer
- **Single Instruction**: All processing units execute the same instruction at any given clock cycle
- **Multiple Data**: Each processing unit can operate on a different data element
- Two varieties: Processor Arrays and Vector Pipelines
- Examples:
  - *Processor Arrays*: Thinking Machines CM-2, MasPar MP-1 & MP-2, ILLIAC IV
  - *Vector Pipelines*: IBM 9000, Cray X-MP, Y-MP & C90, Fujitsu VP, NEC SX-2, Hitachi S820, ETA10
- Most modern computers, particularly those with graphics processor units (GPUs) employ SIMD instructions and execution units.



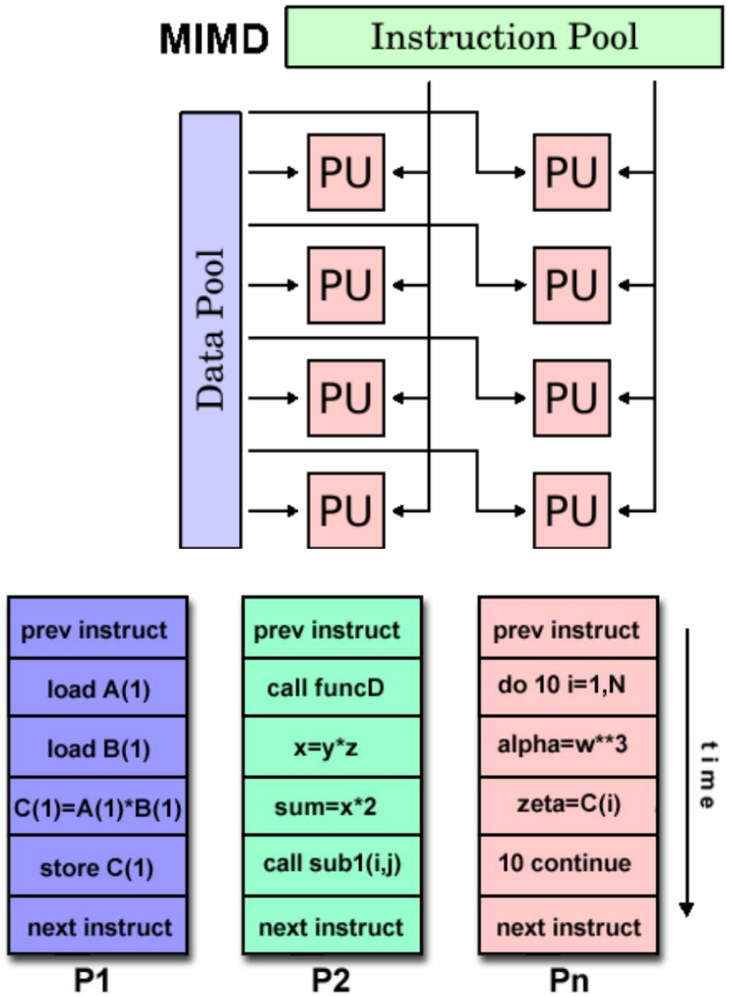
Multiple Instruction, Single Data (MISD)❖

- A type of parallel computer
- **Multiple Instruction**: Each processing unit operates on the data independently via separate instruction streams.
- **Single Data**: A single data stream is fed into multiple processing units.
- Few (if any) actual examples of this class of parallel computer have ever existed.
- Some conceivable uses might be:
  - multiple frequency filters operating on a single signal stream
  - multiple cryptography algorithms attempting to crack a single coded message.



Multiple Instruction, Multiple Data (MIMD)❖

- A type of parallel computer
- **Multiple Instruction**: Every processor may be executing a different instruction stream
- **Multiple Data**: Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Currently, the most common type of parallel computer - most modern supercomputers fall into this category.
- Examples: most current supercomputers, networked parallel computer clusters and "grids", multi-processor SMP computers, multi-core PCs.
- Note: many MIMD architectures also include SIMD execution sub-components



پردازش در گذر زمان


UNIVAC1

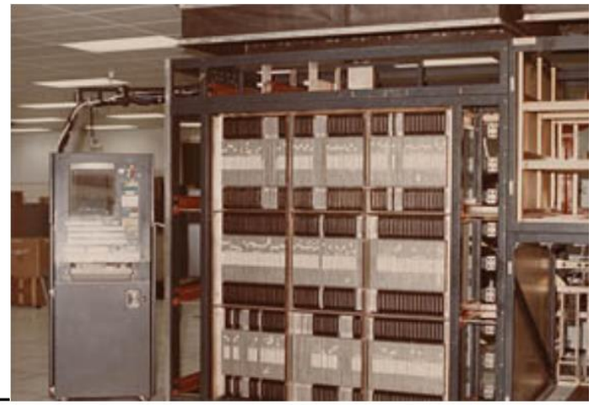

IBM 360


CRAY1


CDC 7600


PDP1

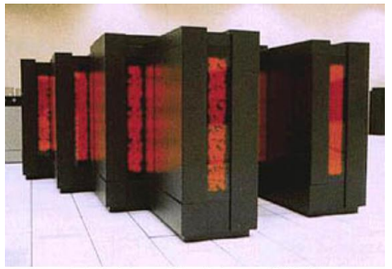

Dell Laptop

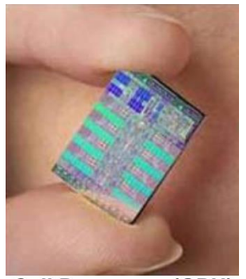

ILLIAC IV


MasPar

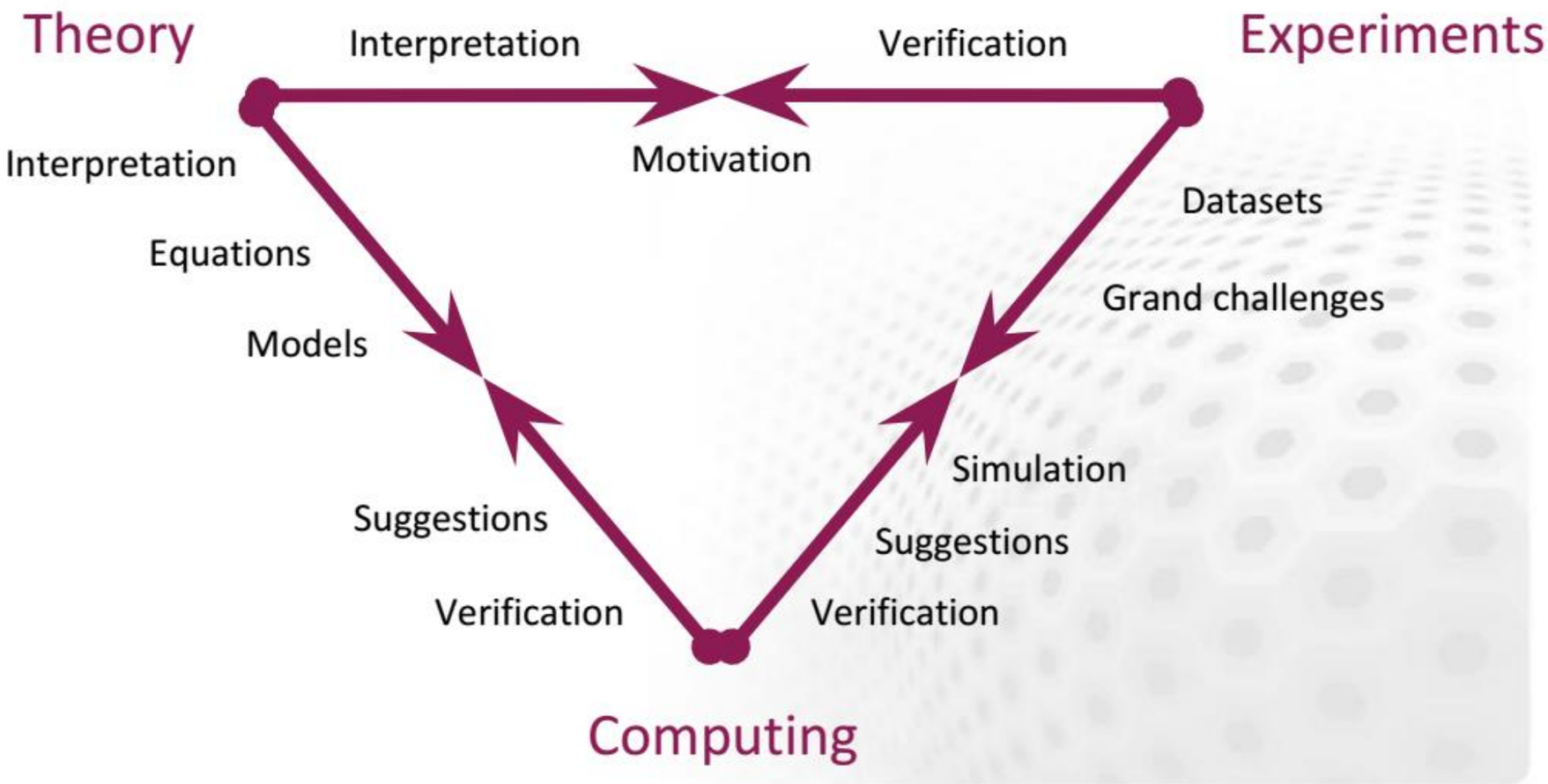

Cray X-MP

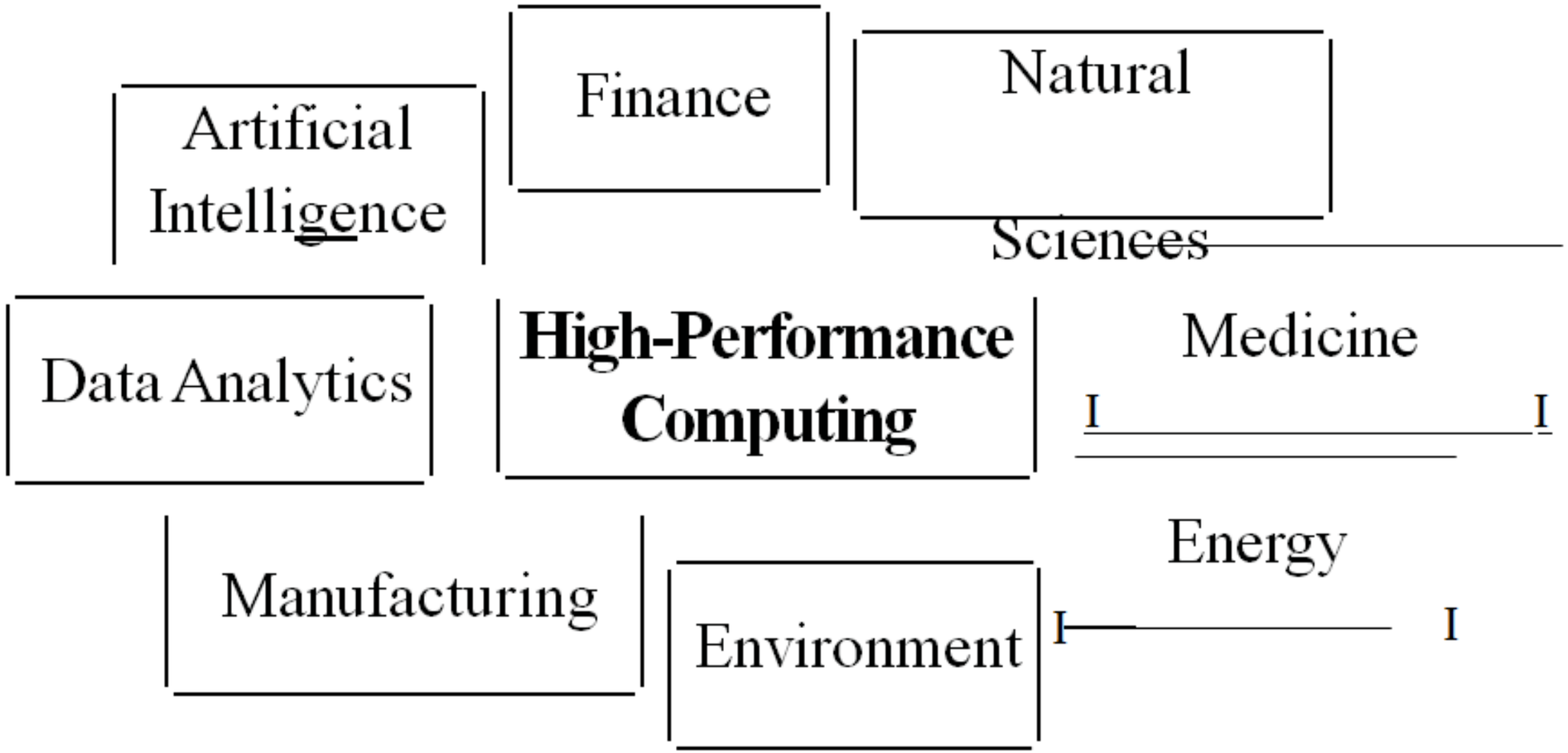

Cray Y-MP


Thinking Machines CM-2

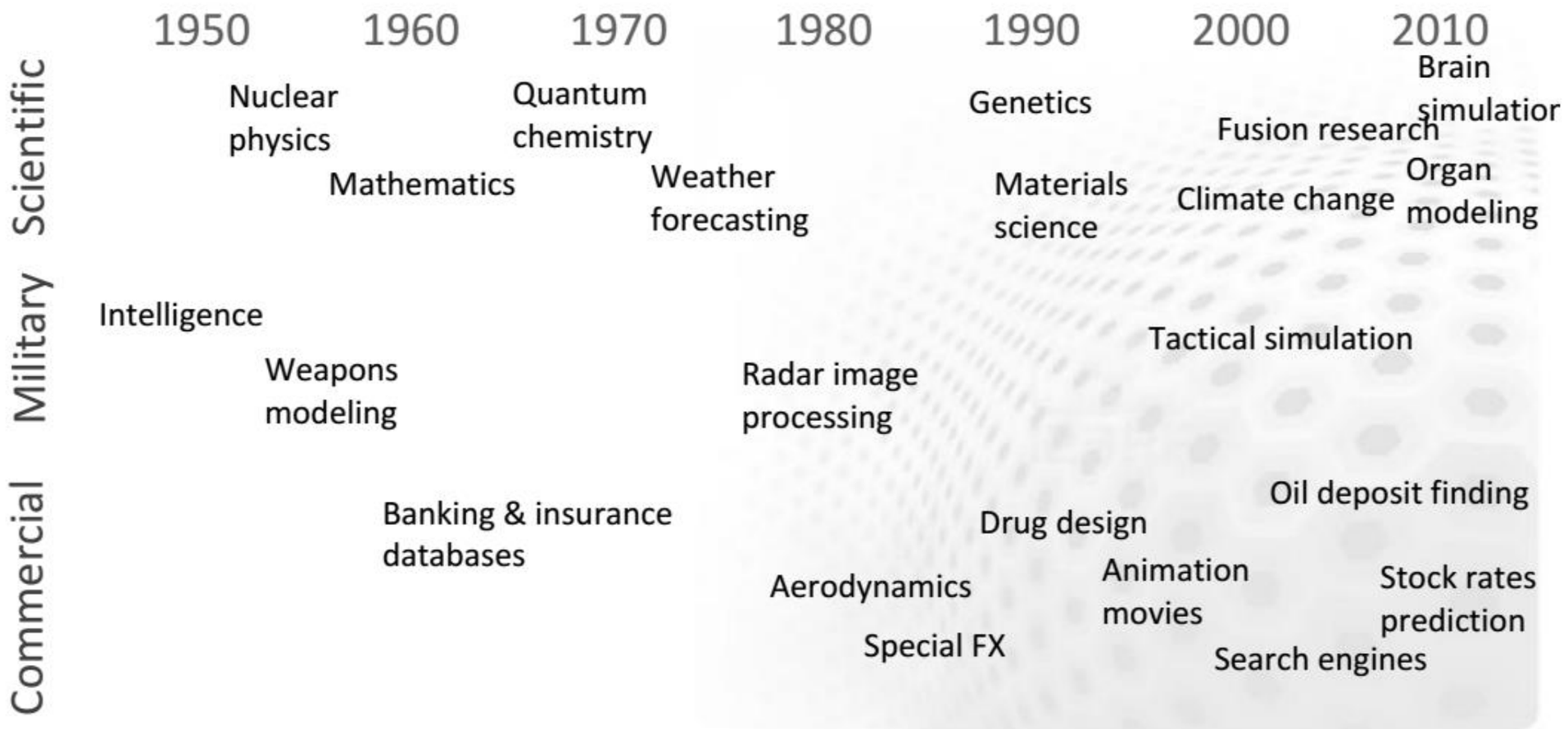

Cell Processor (GPU)

متد علمی رایانش

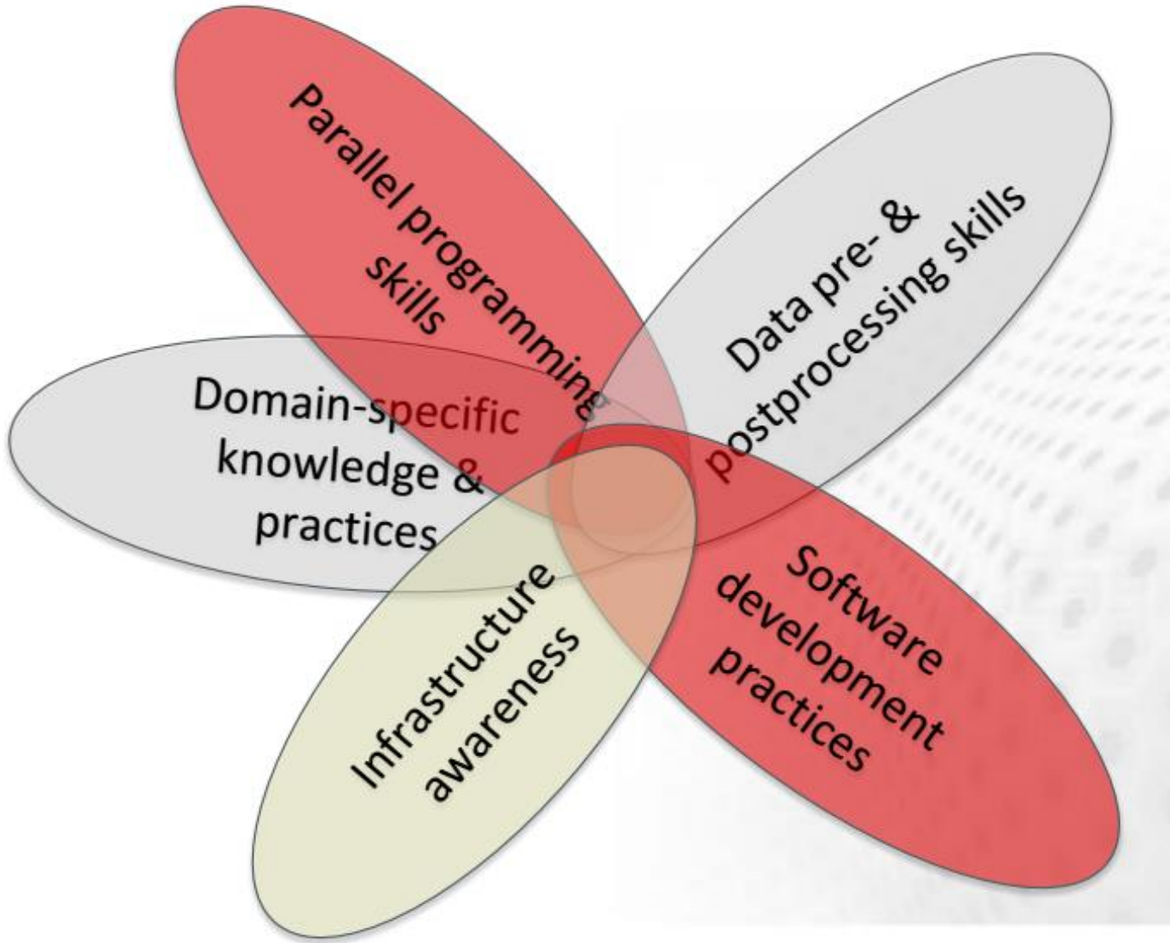


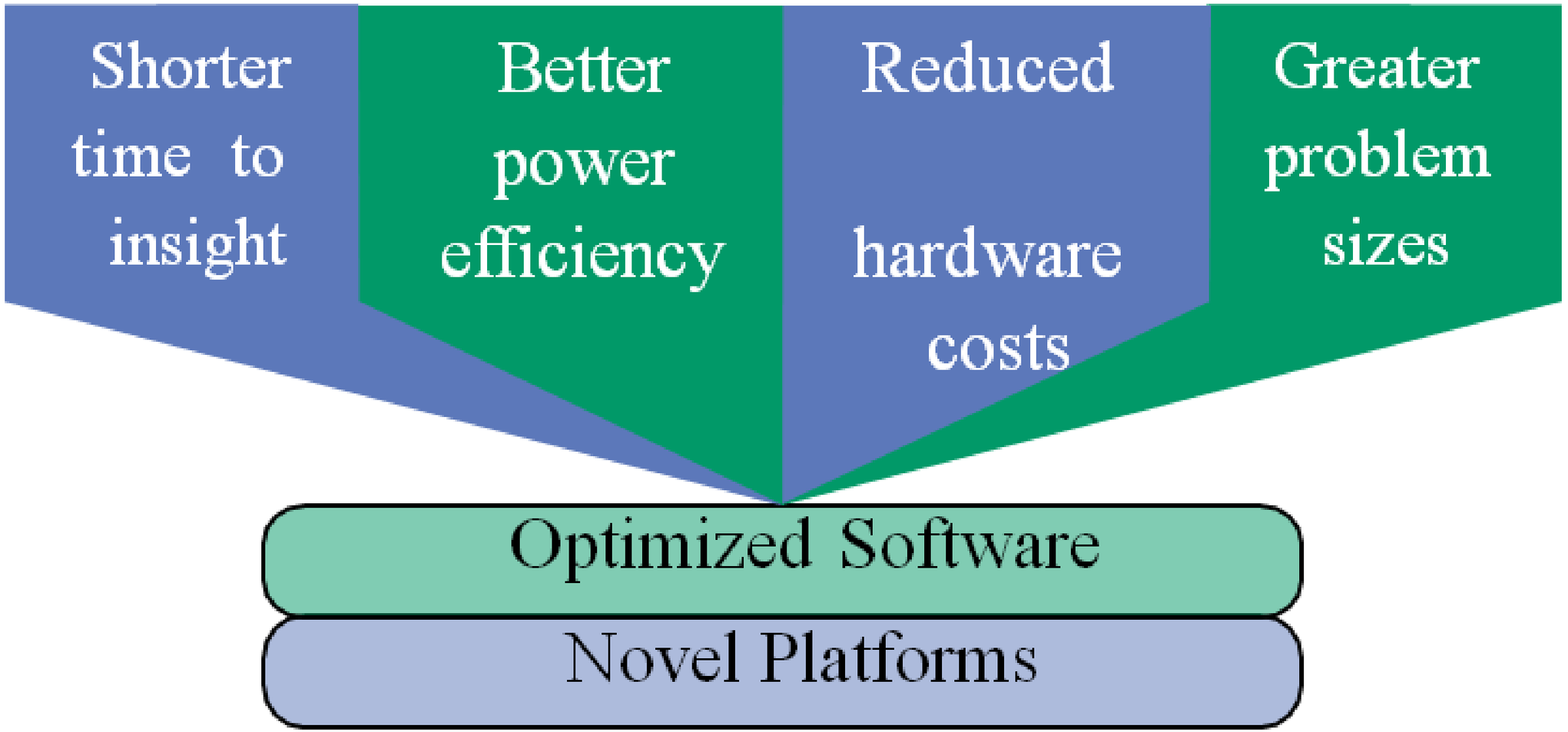

Artificial Intelligence
Finance
Natural Sciences
Data Analytics
High-Performance Computing
Medicine
Manufacturing
Environment
Energy

❖HPC در گذر ایام



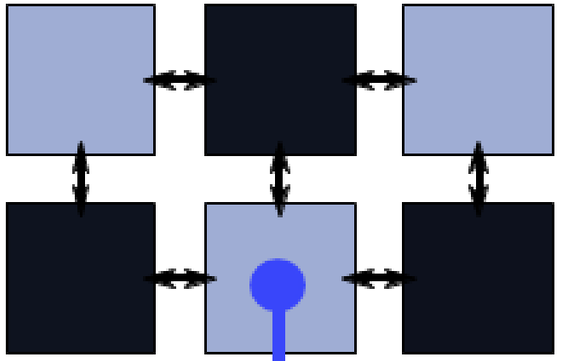
❖ الزامات استفا ده از HPC در محاسبات علمی




Shorter time to insight
Better power efficiency
Reduced hardware costs
Greater problem sizes
Optimized Software
Novel Platforms

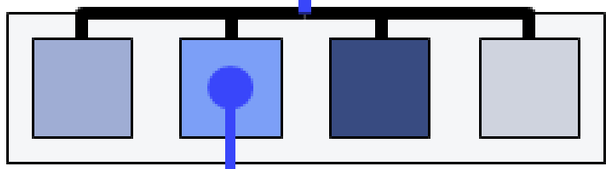
❖لایه های برنامه نویسی موازی

CLUSTER COMPUTING
in distributed memory



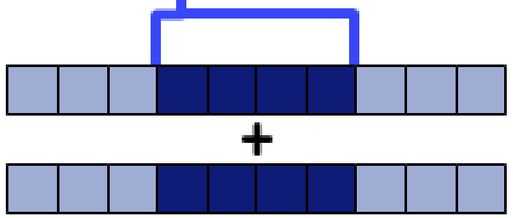
```
MPI_Sendrecv(data, k,
    MPI_DOUBLE, data2,
    ... );
```

MULTITHREADING
in shared memory



```
#pragma omp parallel for
for (j = 0; j < m; j++)
  ComputeSubset(j);
```

VECTORIZATION
of floating-point math



```
#pragma omp simd
for (i = 0; i < n; i++)
  A[i] += B[i];
```

❖ روشهای مختلف موازی سازی مبتنی بر حافظه

- حافظه اشتراکی Shared Memory

  پردازنده های (Core or Processor) مختلف یک کامپیوتر از یک حافظه مشترک برای خواندن و نوشتن داده استفاده می کنند.

  مثال: OpenMP

- حافظه توزیع شده Distributed Memory

  سیستمهای مختلف حافظه اختصاصی خود را دارند و برای تبادل داده بین پردازنده سیستمها از ارسال/دریافت پیام استفاده می شود.

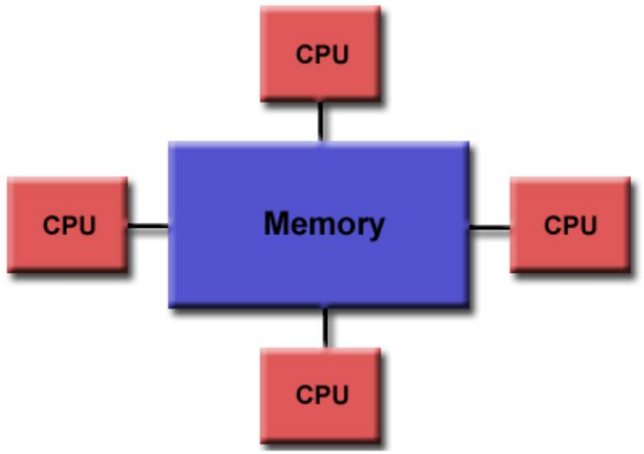
  مثال: MPI

- حافظه ترکیبی Hybrid Memory

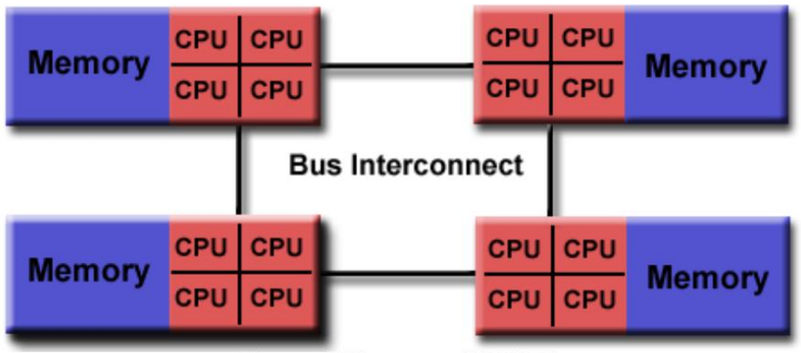
  ترکیب دو روش بالا مورد استفاده قرار می گیرد

  مثال: PGAS

❖حافظه اشتراکی Shared Memory



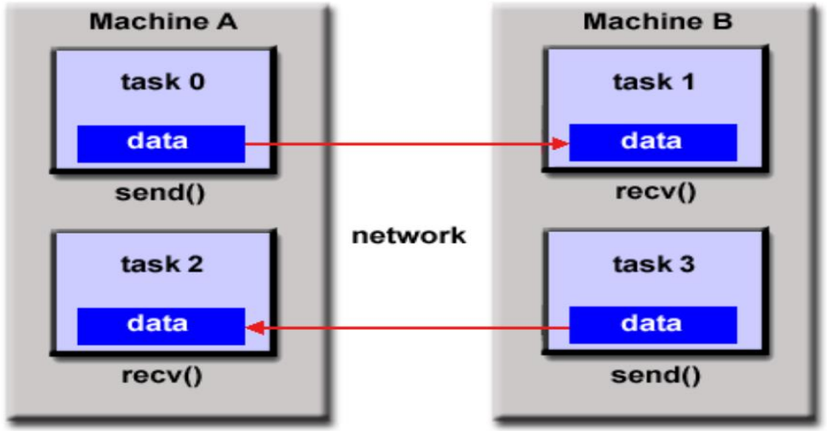
Shared Memory (UMA)

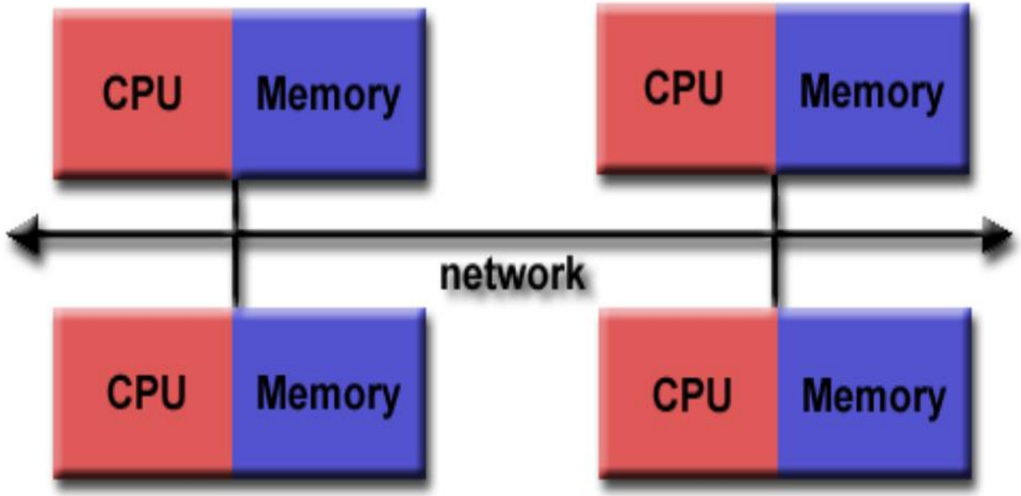


Shared Memory (NUMA)

- مرسوم ترین روش در موازی سازی است
- پردازنده های مختلف مستقل هستند اما از طریق یک حافظه مشترک با هم تبادل داده می کنند
- تغییرات یک داده در حافظه توسط یک پردازنده قابل مشاهده برای سایر پردازنده ها می باشد
- ماشینهادی حافظه مشترک از نظر نوع و سطح دسترسی به دو دسته UMA و NUMA تقسیم می شوند:
- حافظه های UMA عمدتا با SMP(Symmetric Multiprocessor) شناخته می شوند
- در ماشینهای NUMA تمامی پردازنده ها با اتصالات فیزیکی می توانند به حافظه سایر ماشینها دسترسی پیدا کنند
- در NUMA همه پردازنده ها زمان دسترسی یکسانی به حافظه ندارند
- مثال: OpenMP, POSIX Thread, CUDA for GPU

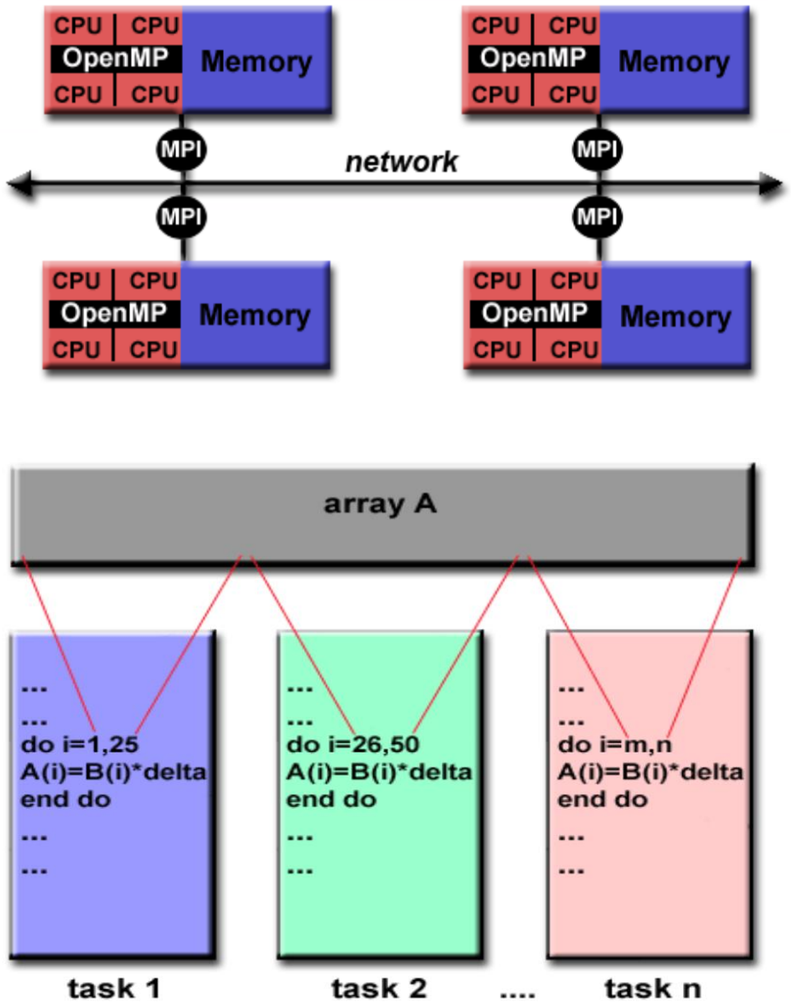
❖حافظه توزیع شده Distributed Memory



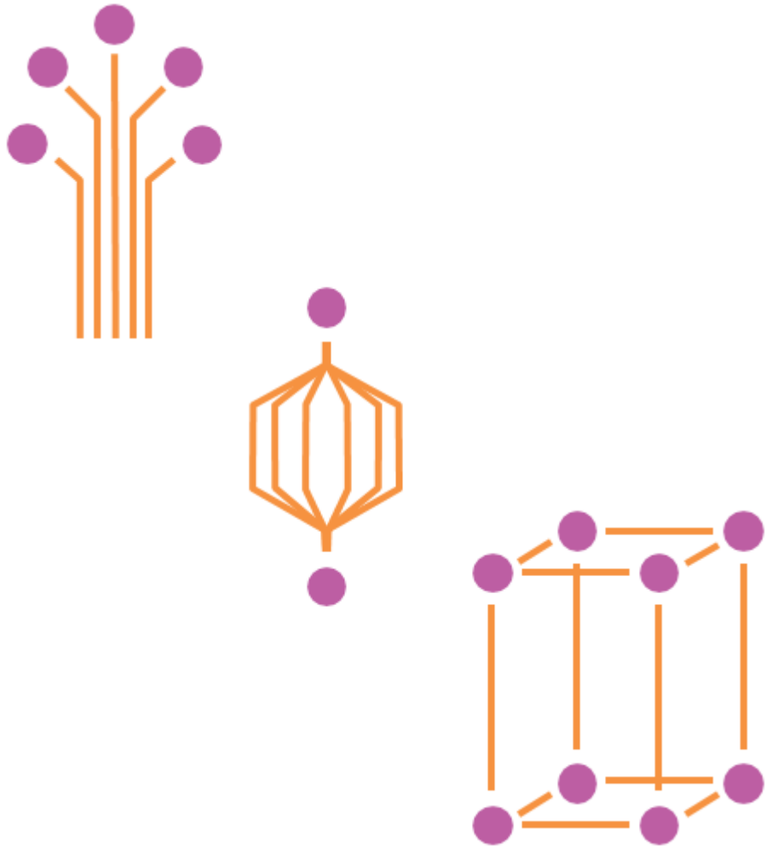


- سیستمهای با حافظه مشترک را به هم متصل می کند
- نیاز به اتصالات شبکه ای به گره های پردازشی وجود دارد
  - مثال: Ethernet, Infiniband, Omni-Path, Myrinet
- هر پردازنده از حافظه اختصاصی خود استفاده می کند و در صورت نیاز به تبادل داده با پردازنده دیگر، از طریق شبکه تبادل صورت می گیرد
- از نظر پردازنده و حافظه مقیاس پذیر است و می توان به سرعت تعداد گره های محاسباتی و حافظه متعلق به هر یک را افزایش/کاهش داد
- از معایب عمده آن نیاز به کد نویسی مجدد نرم افزارهای موجود است(استفاده از کتابخانه های MPI)
- پیشرفت های اخیر در حوزه نرم افزار تا حدود زیادی این عیب را برطرف ساخته است اما تا نقطه ایده آل فاصله دارد
  - مثال: (کتابخانه OpenACC و OpenMP نسخه ۳)

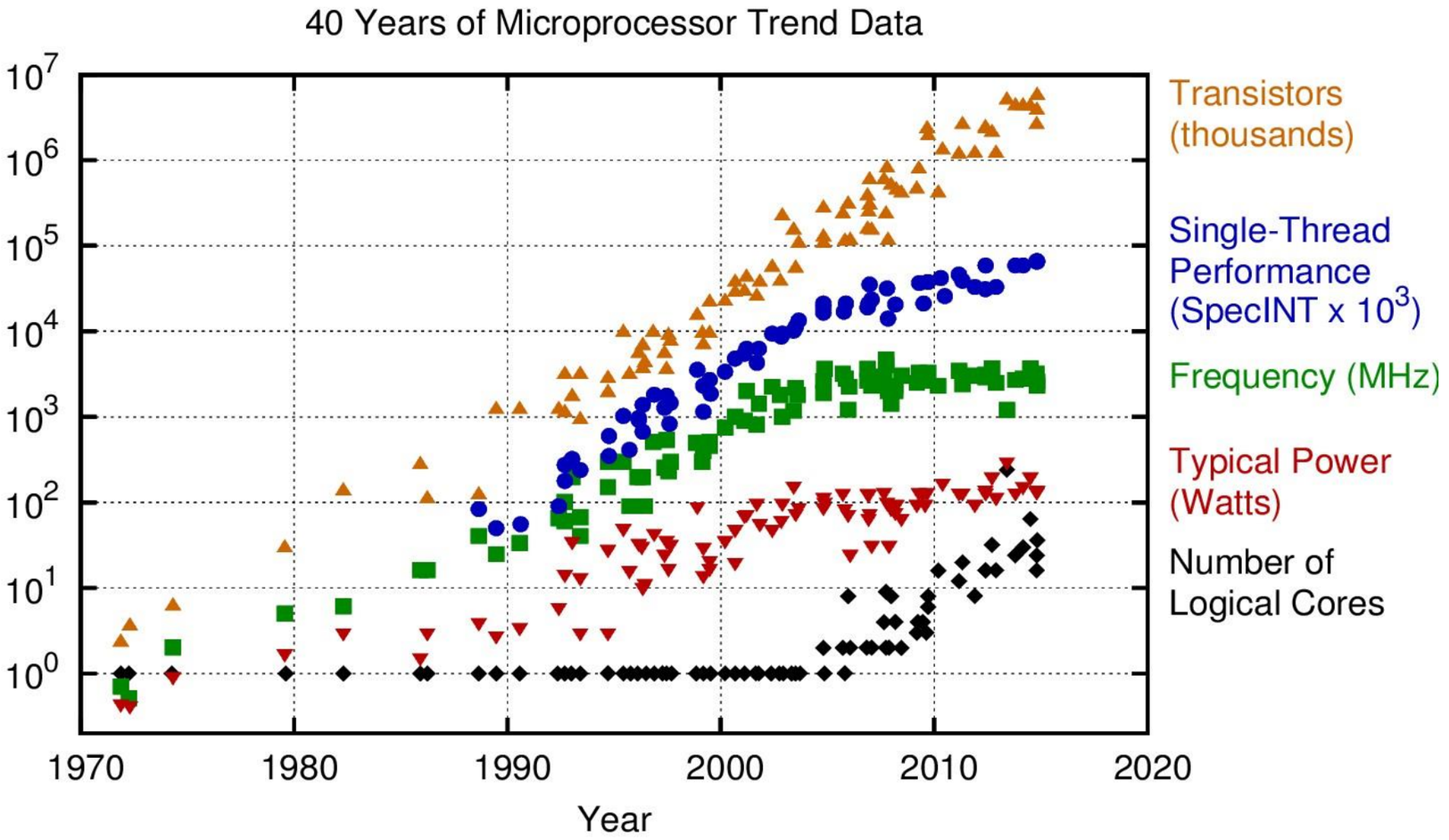
❖ترکیبی Hybrid



- در این مدل دو روش حافظه اشتراکی و توزیع شده با هم ترکیب می شوند
- رایج ترین آنها هم مدل Data Parallel است که داده ها به صورت توزیع شده و اشتراکی با آدرس دهی یکپارچه در RAM کلیه گره های محاسباتی قرار می گیرند
  - مثال : PGAS(Partitioned Global Address Space)
  - Coarray Fortran, Unified Parallel C(UPC)
  - Global Array, X10, Chapel
- آدرس دهی به حافظه یکپارچه است
- در صورتی که پردازنده های مبداء و مقصد در یک سیستم باشند از حافظه اشتراکی(OpenMP) و در غیر اینصورت از حافظه توزیع شده (MPI) استفاده می کنند



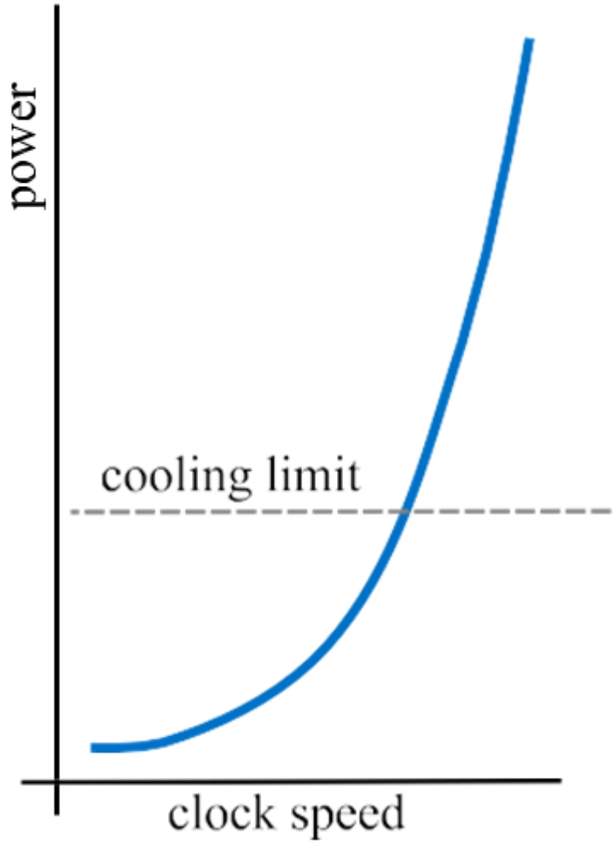
چطور می توان نرم افزارها را سریعتر کرد؟



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Source: karlrupp.net

❖ سرعت پردازنده ها مسئله PowerWall



Overclocking?

Free cooling?

راه حلهای خنک کنندگی مرسوم برای پردازنده های با سرعت بالا عملی نبوده و یا گران هستند

❖ انواع روشهای موازی سازی مرسوم

- ❑ Bit Level Parallelism
- ❑ Instruction Level Parallelism(ILP)
  1. Explicitly Parallel Instruction Computing (EPIC)
  2. Out of Order Execution/Register Renaming
  3. Speculative Execution
  4. Vectorization
- ❑ Threading/Multi-threading

❖روش Bit Level Parallelism

با توسعه تکنولوژی VLSI اندازه کلمه یا Word در پردازنده ها افزایش پیدا کرده و توانایی بالاتری در ارتقاء کارایی از خود نشان داده اند.

برای مثال پردازنده های ۳۲ بیتی می توانند دو عدد ۳۲ بیتی را با سرعت بیشتری نسبت به پردازنده ۱۶ بیتی با هم جمع کنند، چرا که یک پردازنده ۱۶ بیتی برای جمع دو عدد ۳۲ بیتی می بایست آن را به دو بخش بالایی و پایینی تقسیم کند که این امر موجب ایجاد سربار محاسباتی می شود.

❖روش Instruction Level Parallelism(ILP)

در این روش کامپایلر سعی می کند دستوراتی را که می توانند با یکدیگر اجرا شوند و وابستگی داده ای به یکدیگر ندارند در یک Cycleیا Clockاجرا نماید.

- Explicitly Parallel Instruction Computing (EPIC)

  در این روش در یک Cycleبین دو تا ۱۶ دسترالعمل قابل اجرا است که هر چه تعداد دستورات غیروابسته به هم کمتر باشد درصد بالاتری از موازی سازی را ایجاد می کند.

- Out of Order Execution/Register Renaming

  در این روش امکان اجرای دستورات بیشتری در یک Cycleوجود دارد که با تغییر نام ثبات ها می توان به این حالت دست یافت.

- Speculative Execution

  اجرای تمام یا بخشی از دستورات صرف نظر از اینکه زمان اجرای آن فرا رسیده است یا خیر.

- Vectorization

  این یکی از حالات خاص از مدل (SIMD (Simple Instruction Multiple Dataاقای Flynnاست که در آن یک دستور می تواند روی چندین داده به صورت همزمان کار کند. چیپ ست های x86دارای معماری مجموعه دستورالعمل های (SSE Streaming SIMD Extension)هستند. برداری سازی به صورت خودکار پس از انتخاب پارامتر مناسب زمان کامپایل توسط کامپایلر انجام می شود.

تعادل در روشهای پایپ لاین و ILP در موازی سازی

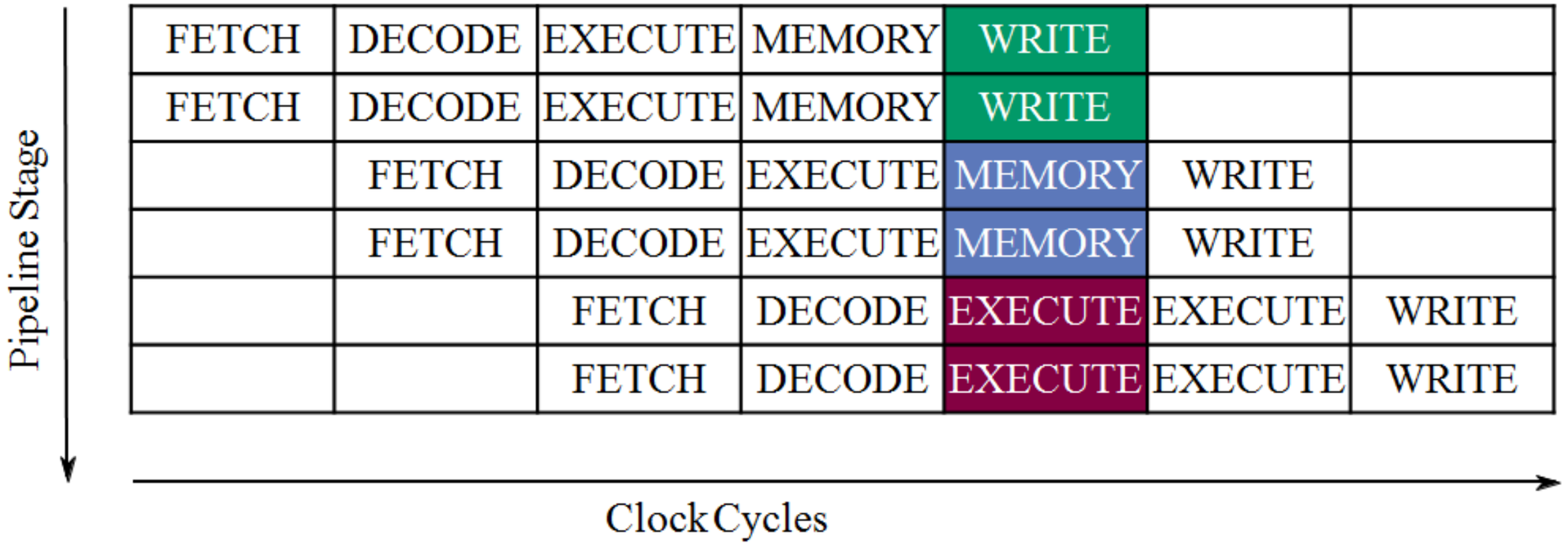
FETCH	DECODE	EXECUTE	MEMORY	WRITE		
	FETCH	DECODE	EXECUTE	MEMORY	WRITE	
		FETCH	DECODE	EXECUTE	MEMORY	WRITE
			FETCH	DECODE	EXECUTE	MEMORY
				FETCH	DECODE	EXECUTE
					FETCH	DECODE

Pipeline Stage

Clock Cycles

زمانی که تعداد پایپ لاین ها افزایش می یابد، تداخل بین دستورالعملها بیشتر می شود

❖ تعادل در اجرای Superscalar و ILP در موازی سازی



جستجوی خودکار برای دستورالعملهای مستقل نیازمند منابع بیشتری است

❖ مسئله حافظه در روش Out-of-Order

In-Order

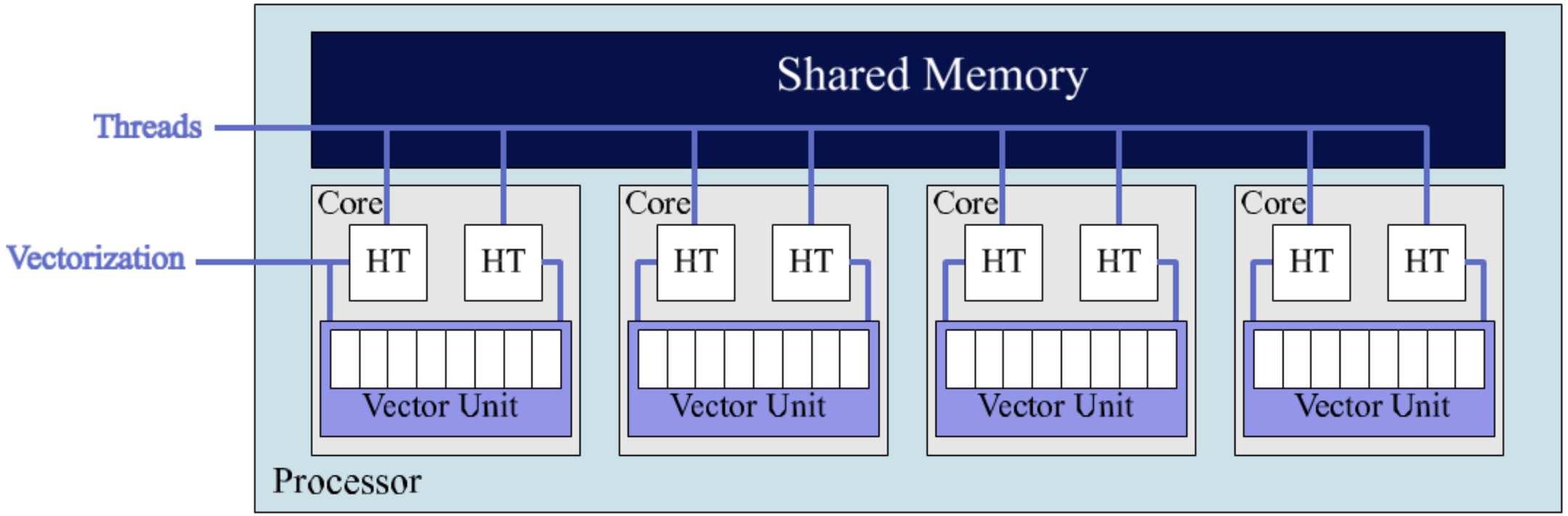
FETCH	DECODE	EXECUTE	MEMORY	MEMORY	MEMORY	WRITE
	FETCH	DECODE	STALL	STALL	STALL	EXECUTE
		FETCH	STALL	STALL	STALL	DECODE

Out-of-Orer

FETCH	DECODE	EXECUTE	MEMORY	MEMORY	MEMORY	WRITE
	FETCH	DECODE	EXECUTE	WRITE		
		FETCH	DECODE	EXECUTE	WRITE	

بوسیله الگوی دسترسی به داده ها در برنامه های کاربردی محدود شده است

❖ موازی سازی : هسته ها و بردارها



هسته زیاد و برداری سازی:

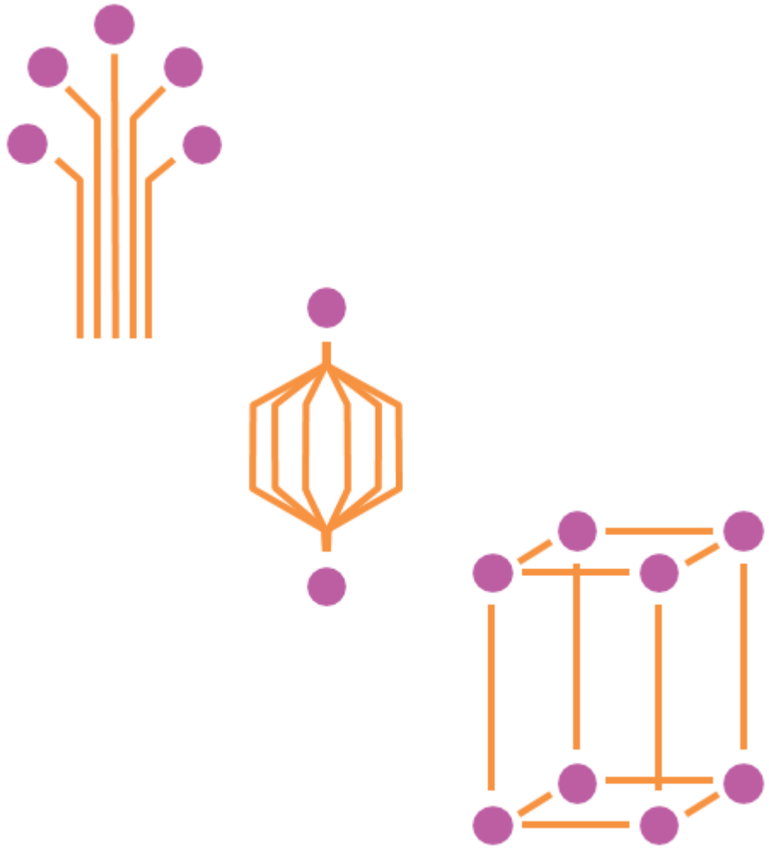
فرصت رشد بدون محدودیت اما نه به صورت خودکار را فراهم می کند

❖ موازی سازی در مسیر رو به جلوست

- مسئله مصرف انرژی Power wall
- مسئله موازی سازی در سطح دستورات ILP wall
- مسئله حافظه Memory wall

نمایش باید ادامه پیدا کند!!

- سخت افزارها از طریق موازی سازی در حال تکامل هستند
- نرم افزار باید عقب بماند!



معماری اینتل

❖ پلترفم های محاسباتی مبتنی بر معماری اینتل

General-Purpose Processors

Intel® Xeon®
Intel® Core™
Intel® Atom™, ...

Specialized Processors

Intel® Xeon Phi™ processors and coprocessors

Computing Accelerators

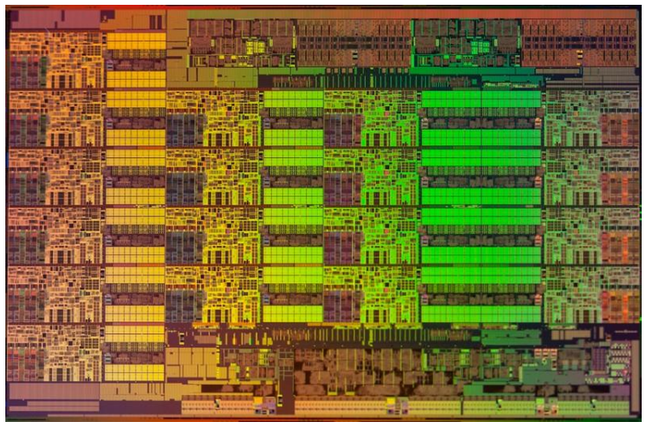
Intel® VCA (x86)
Intel® Nervana™ Platform
Intel® DLIA™ (FPGAs)

Network Interconnects

Intel® Omni-Path™ Architecture

Intel Xeon Processors ❖

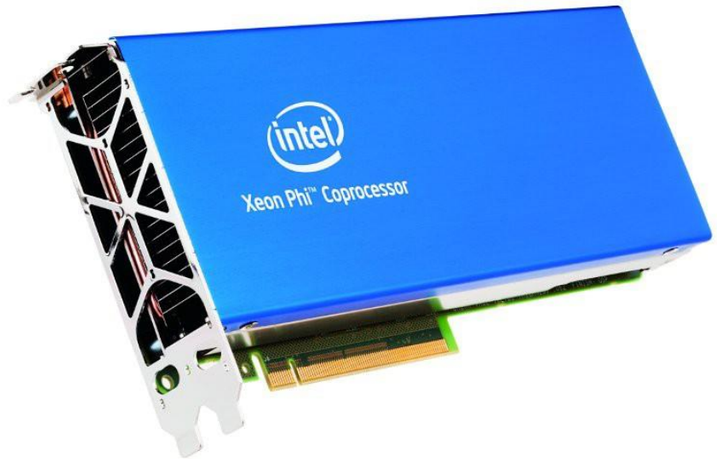
- 1-, 2-, 4-way
- General-purpose
- Highly parallel (44 cores*)
- Resource-rich
- Forgiving performance
- Theor. ~ 1.0 TFLOP/s in DP*
- Meas. ~ 154 GB/s bandwidth*

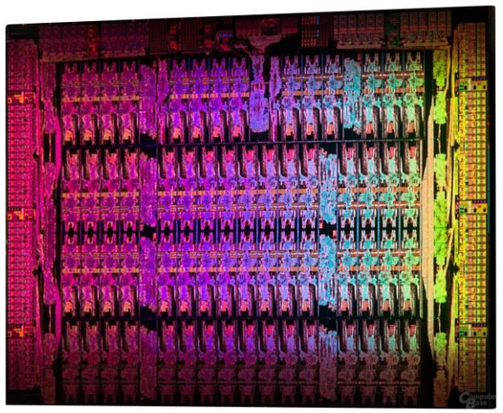


* 2-way Intel Xeon processor, Broadwell architecture (2016), top-of-the-line (e.g., E5-2699 V4)

Intel Xeon Phi Processors (1st Gen) ❖

- PCIe add-in card
- Specialized for computing
- Highly-parallel (61 cores*)
- Balanced for compute
- Less forgiving
- Theor. ~ 1.2 TFLOP/s in DP*
- Meas. ~ 176 GB/s bandwidth*

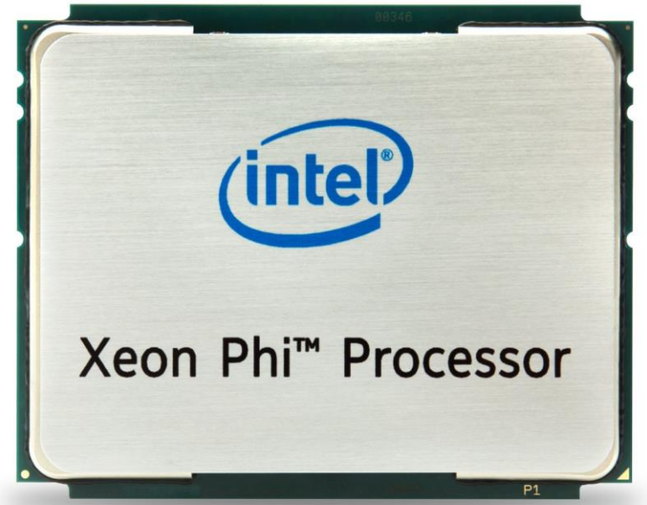


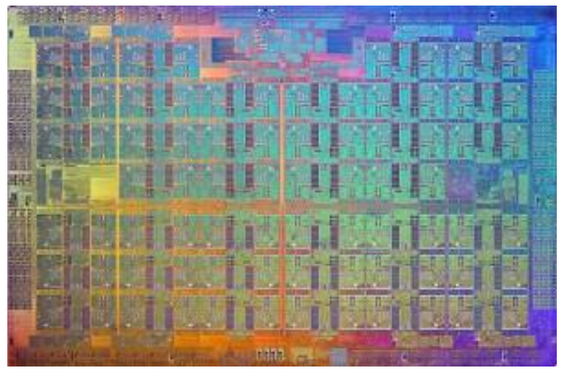


* Intel Xeon Phi coprocessor, Knighs Corner architecture (2012), top-of-the-line (e.g., 7120P)

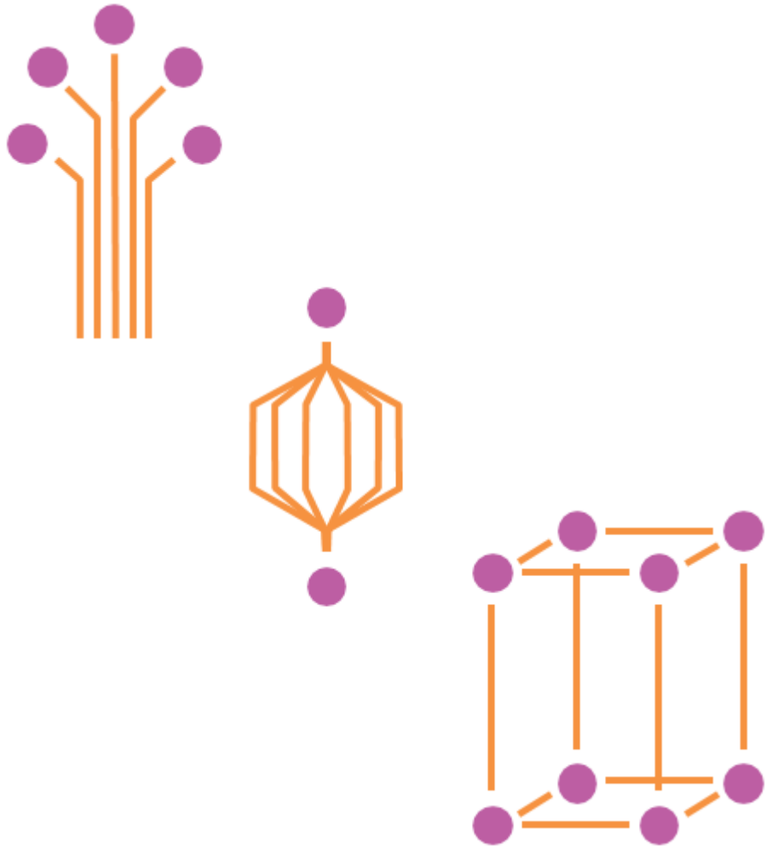
Intel Xeon Phi Processors (2nd Gen) ❖

- Bootable or PCIe add-in card
- Specialized for computing
- Highly-parallel (72 cores*)
- Balanced for compute
- Less forgiving than Xeon
- Theor. ~ 3.0 TFLOP/s in DP*
- Meas. ~ 490 GB/s bandwidth*



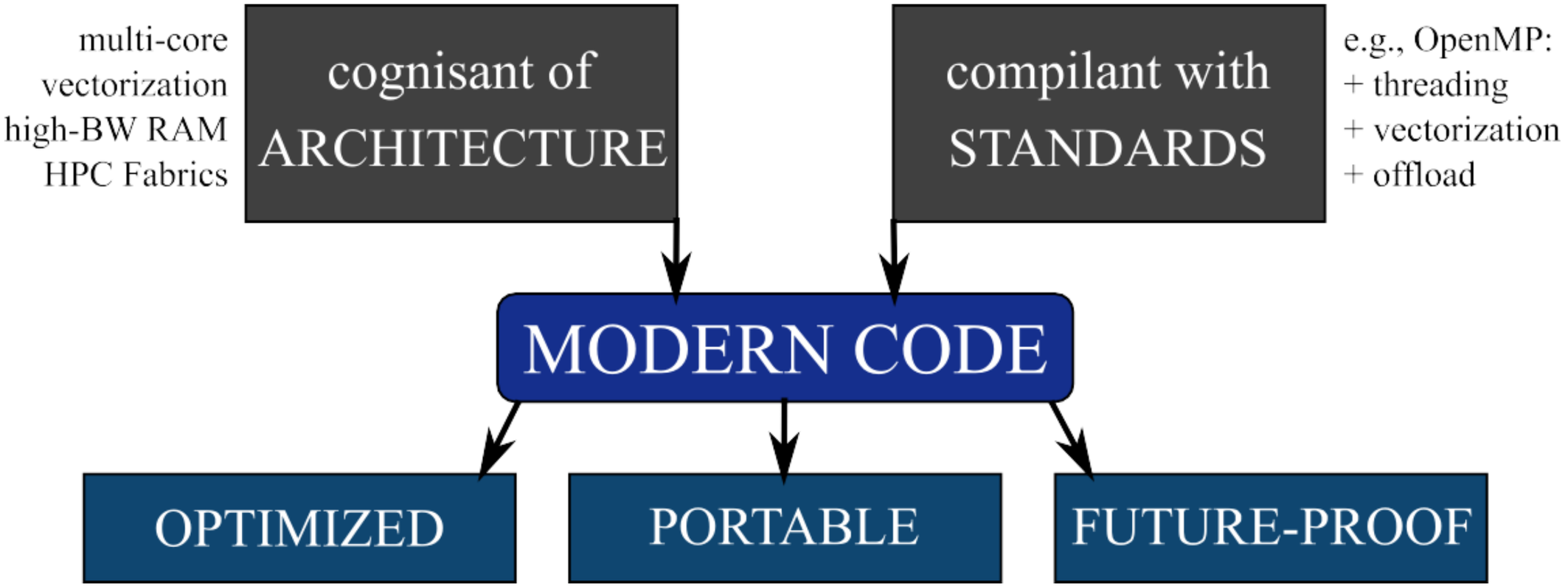


* Intel Xeon Phi processor, Knighs Landing architecture (2016), top-of-the-line (e.g., 7290P)

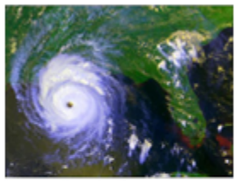


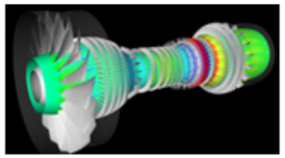
کدهای مدرن

یک کد برای همه پلتفرم ها



❖ رایانش و محاسبات در علم و مهندسی





❖چه کسانی به HPC نیاز دارند؟

به شکل تاریخی رایانش موازی با عنوان "the high end of computing" شناخته می شود و از آن برای حل و مدل سازی مسائل سخت در بسیاری از حوزه های مهندسی و علم استفاده می شود.

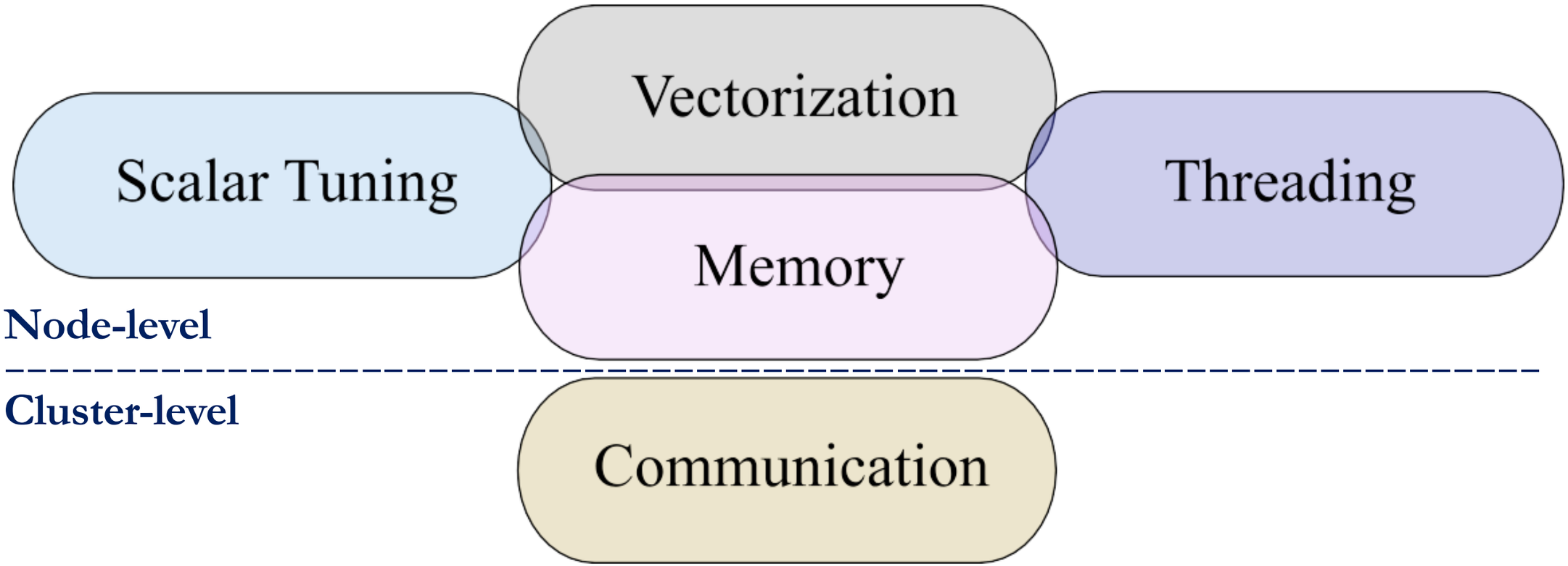
- Atmosphere, Earth, Environment
- Physics - applied, nuclear, particle, condensed matter, high pressure, fusion, photonics
- Bioscience, Biotechnology, Genetics
- Chemistry, Molecular Sciences
- Geology, Seismology
- Mechanical Engineering – from prosthetics to spacecraft
- Electrical Engineering, Circuit Design, Microelectronics
- Computer Science, Mathematics
- Defense, Weapons

❖چه کسانی به HPC نیاز دارند؟ – ادامه

امروزه حوزه های تجاری نیز برای بسیاری از تجزیه و تحلیلهای خود به این بستر محتاج شده اند. رشد زیاد حجم داده ها باعث شده است که نیاز به پردازش بیشتر حس شود. از جمله این حوزه ها می توان به موارد زیر اشاره کرد:

- "Big Data", databases, data mining
- Oil exploration
- Web search engines, web based business services
- Medical imaging and diagnosis
- Pharmaceutical design
- Financial and economic modeling
- Management of national and multi-national corporations
- Advanced graphics and virtual reality, particularly in the entertainment industry
- Networked video and multi-media technologies
- Collaborative work environments

مناطق بهینه سازی



تجربه نوسازی و بهینه سازی کدهای مرسوم و معمول