INFO-H-515:
BIG DATA ANALYTICS

Introduction to big data analytics

Gianluca Bontempi

Machine Learning Group
Boulevard de Triomphe - CP 212
http://mlg.ulb.ac.be

INFO-H-515: SECOND PART

- Second part of the course Big Data : Distributed Data Management and Scalable Analytics
- Required
  - first part of the course (throughput, latency, distributed computing, MapReduce, Lambda Architectures, Spark, Spark streaming, scalability)
  - foundations of Machine Learning (INFOF422)
  - programming skills in Python, Spark
- **From data management perspective to analytics** perspective.
- Exam:
  1. Some oral questions in the UV exam.
  2. Project on LLM.

INTRODUCTION

FROM ML TO BIG DATA ANALYTICS

- What should we keep in mind about the INFO-F-422 course ”Statistical foundations of machine learning”?
- Is an additional course really necessary ?
- Is ”big data” just an hype?
- What is new? what is not new?

WHEN BIG IS BIG?

- Pragmatic "moving target" definition: When either the data is **too large to fit** on a single machine or it would simply take **too long to perform** that computation on a single machine.
- Data is big when **conventional single machine computing is no more feasible** or effective.
- Conventional comuputing limitation:
  - data processing algorithms with wide data dependencies suffer because network transfer rates are orders of magnitude slower than memory accesses.
  - as the number of machines working on a problem increases, the probability of a failure increases.
- Are all analytics problems big data problems? of course not.
- There are still many companies/business/disciplines claiming big data problems though those are simple conventional analytics problems.

WHEN BIG IS BIG?

Dataset type	Fits in RAM?	Fits on local disk?	
Small dataset	Yes	Yes	conventional
Medium dataset	No	Yes	parallelism, database
Big dataset	No	No	cluster, map/reduce

Libraries like Pandas are not equipped to deal with data not fitting in the RAM.

WHAT IS NOT NEW?

Big data analytics remains an instance of a **statistical modeling problem**, which aims at inferring estimators/predictors from observations.
Useful notions to keep in mind:

- Estimator as function of data.
- Supervised/unsupervised.
- Data preparation.
- Dimensionality reduction.
- Generalization, bias/variance, underfitting/overfitting.
- Model selection and assessment.
- Iterative refinements

WHAT IS NEW?

Special focus on big data issues

- **Volume**: in terms of variables and/or samples. It calls for scalable storage and a distributed approach to learning.
- **Velocity**: it is the rate at which data flows into. Streaming data tasks, online learning.
- **Variety**: beyond conventional numeric measurements, heterogenous sources (blog posts, tweets, social network interactions, photos, logs).
- Beyond accuracy: performance (e.g. throughput, latency).
- Distribution of the computation: scalability.
- New programming languages, paradigms and tools.

WHAT IS NEW?

Many of the basic assumptions made in single-node systems are no more valid.

- **Partitioning** of data across many nodes: algorithms with data dependencies suffer from the fact that network transfer rates are orders of magnitude slower than memory accesses.
- **No shared memory**
- As the number of machines working on a problem increases, the probability of a **failure** increases.
- **Programming** paradigm adapted to underlying system which makes easy to write highly parallel code.
- Most literature is ad-hoc (specific learning machine) and provides no general approach for distributing machine learning .

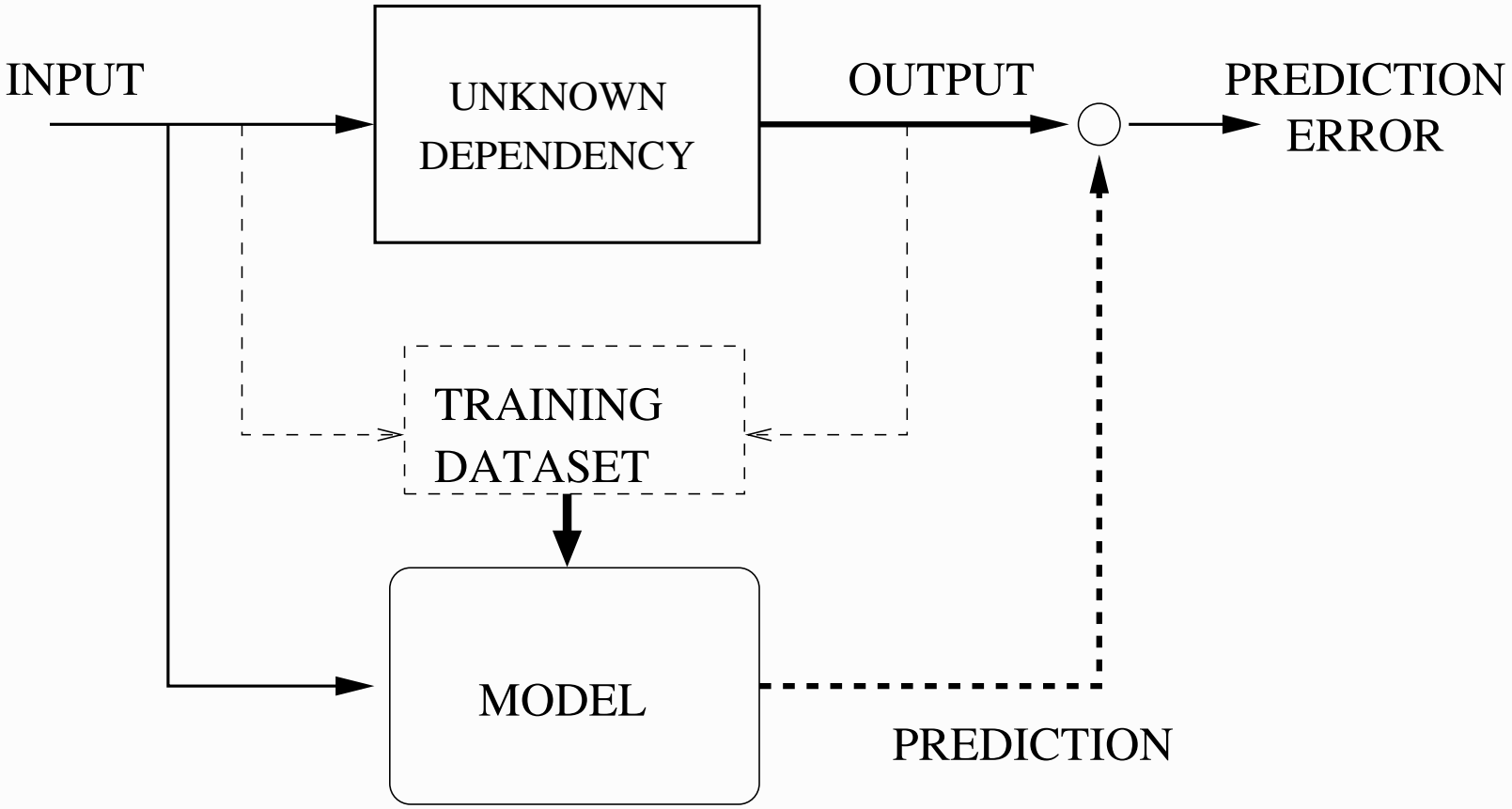
ML CRASH COURSE

SUPERVISED VS UNSUPERVISED TASKS

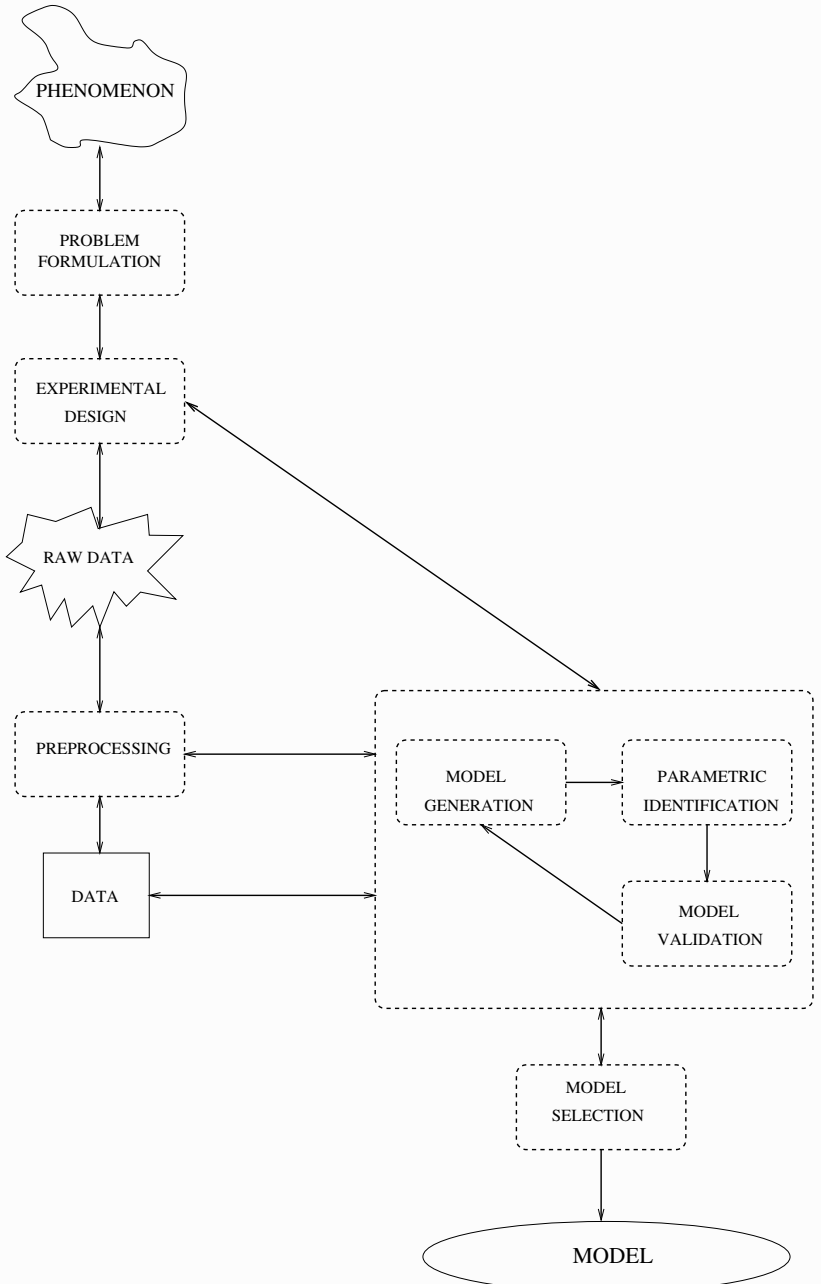
- Supervised
  - Regression
  - Classification
  - Time series forecasting
- Unsupervised
  - Clustering
  - Outlier detection
  - Density estimation

For en extended introduction to the topic refer to [**?**].

SUPERVISED LEARNING SETTING



MACHINE LEARNING PIPELINE



PREPROCESSING

This step is more and more important and time consuming as data size grows.

- Data collection
- Data cleaning
- Missing data management
- Data formatting

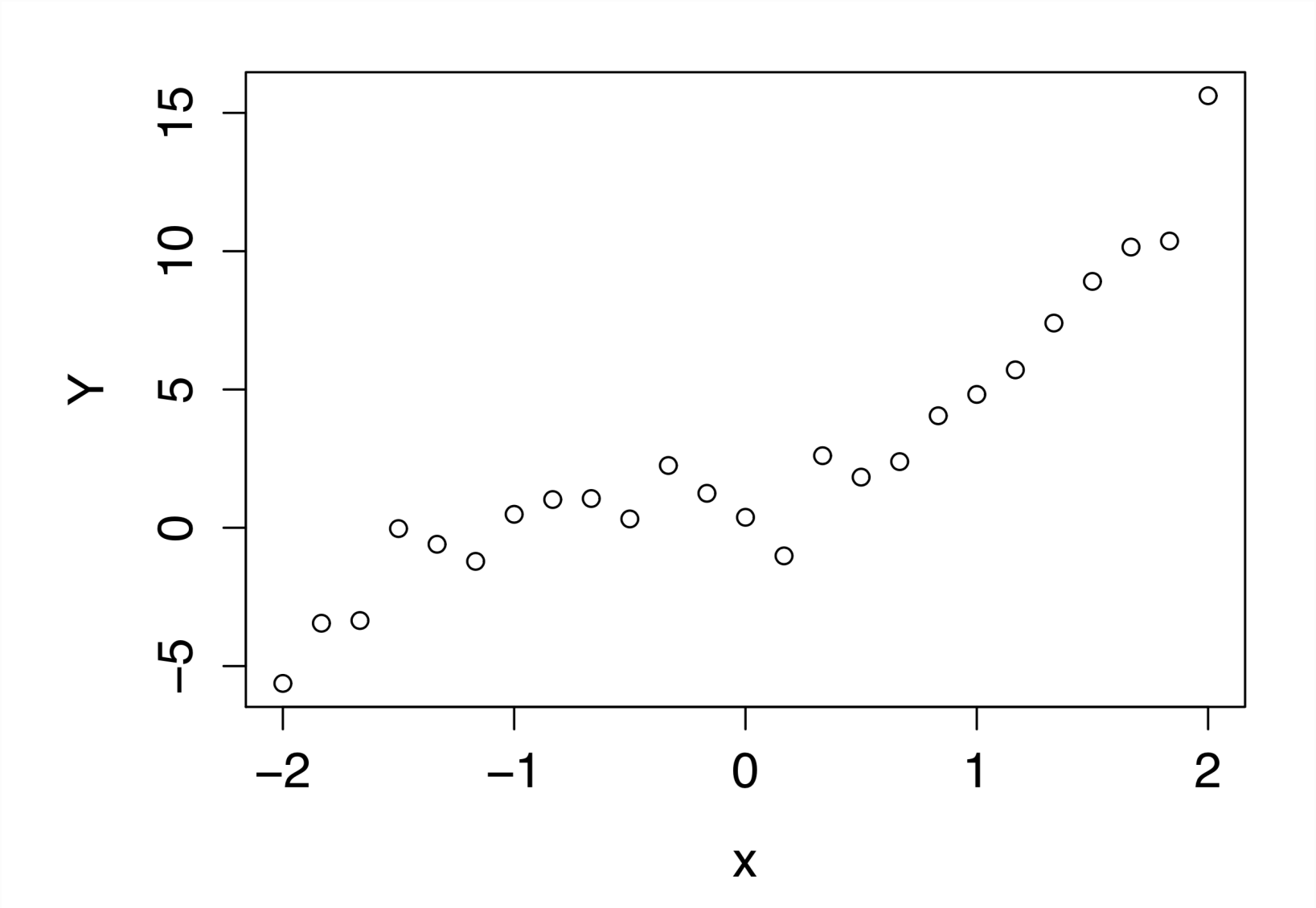
TABULAR DATA FORMATS

We assume that training dataset D_N can be formatted as

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Nn} \end{bmatrix}$$

where N is the number of observations and n is the number of features.

SIMPLE REGRESSION TASK



UNDERFITTING/OVERFITTING

UNDERFITTING/OVERFITTING

UNDERFITTING/OVERFITTING

LEARNING PROCEDURE

A learning procedure [**?**] aims at:

1. choosing a parametric family of hypothesis $h(x, \alpha)$ which contains or gives good approximation of the unknown function f (**structural identification**).
2. within the family $h(x, \alpha)$, estimating (on the basis of D_N) the parameter α_N which minimises the training error (**parametric identification**).

Two nested loops:

1. an external structural identification loop through different model structures
2. an inner parametric identification loop searching for the best parameter vector within the family structure.

PARAMETRIC IDENTIFICATION

Empirical Risk Minimization (ERM):

$$\alpha_N = \alpha(D_N) = \arg\min_{\alpha \in \Lambda} \widehat{MISE}_{emp}(\alpha)$$

It minimizes the **empirical risk or training error**

$$\widehat{MISE}_{emp}(\alpha) = \frac{\sum_{i=1}^{N} (y_i - h(x_i, \alpha))^2}{N}$$

on the basis of D_N.
Training error is a biased (i.e. optimistic) estimation of the real generalization error (MISE).

VALIDATION TECHNIQUES

How to measure MISE in a reliable way on a finite dataset?

- **Testing:** testing sequence independent of D_N and with the same distribution.
- **Holdout:** partition of D_N into two mutually exclusive subsets: training set D_{tr} and test set $D_{N_{ts}}$.
- k**-fold Cross-validation:** D_N is randomly divided into k mutually exclusive test partitions of approximately equal size. The cases not found in each test partition are used for training the hypothesis which is then tested on the partition itself.

MODEL SELECTION

- Final choice of the model structure after model generation and validation.
- Two approaches:
  1. winner-takes-all approach: choose the model structure that minimize an accurate estimate of the generalization error (e.g. cross-validation)
  2. combination of estimators approach: combination (e.g. averaging) of different models (e.g. different orders, different family).

MODEL COMBINATION

- The winner-takes-all approach is intuitively the approach which should work the best.
- Accuracy of the final model can be improved not by choosing the model structure which is expected to predict the best but by creating a model whose output is the combination of the output of different models.
- Rationale: any chosen hypothesis $h(\cdot, \alpha_N)$ is only an estimate of the real target and, like any estimate, is affected by a bias and a variance term.
- See in [**?**] theoretical results on the combination of estimators.
- Bagging is a well-known example of model combination.

BAGGING (BOOTSTRAP AGGREGATING)

- Consider a dataset D_N and a learning procedure to build an hypothesis α_N from D_N.
  i
- A set of B repeated bootstrap samples $D_N^{(b)}$, $b = 1, \ldots, B$ are taken from D_N.
- A model $\alpha_N^{(b)}$ is built for each $D_N^{(b)}$.
- A final predictor is built by aggregating the B models $\alpha_N^{(b)}$.
- In the regression case the bagging predictor is

$$h_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} h(x, \alpha_N^{(b)})$$

- In the classification case a majority vote is used.

FEATURE SELECTION

There are many potential benefits of feature selection:

- facilitating data visualization and data understanding,
- reducing the measurement and storage requirements,
- reducing training and utilization times of the final model,
- defying the curse of dimensionality to improve prediction performance.

FEATURE SELECTION STRATEGIES

1. **Filter methods:** preprocessing methods. They attempt to assess the merits of features from the data, ignoring the effects of the selected feature subset on the performance of the learning algorithm. Examples are methods that select variables by ranking them through compression techniques (like PCA or clustering) or by computing correlation with the output.
2. **Wrapper methods:** these methods assess subsets of variables according to their usefulness to a given predictor. The method conducts a search for a good subset using the learning algorithm itself as part of the evaluation function.
3. **Embedded methods:** selection as part of the learning procedure and are usually specific to given learning machines. Examples are classification trees, random forests, and methods based on regularization techniques (e.g. lasso)

OTHER ISSUES

- Unbalancedness
- Non stationarity, concept drift
- Semi supervised learning, active learning

BIG DATA ANALYTICS APPLICATIONS

BIG DATA ANALYTICS APPLICATIONS

Some examples [**?**]:

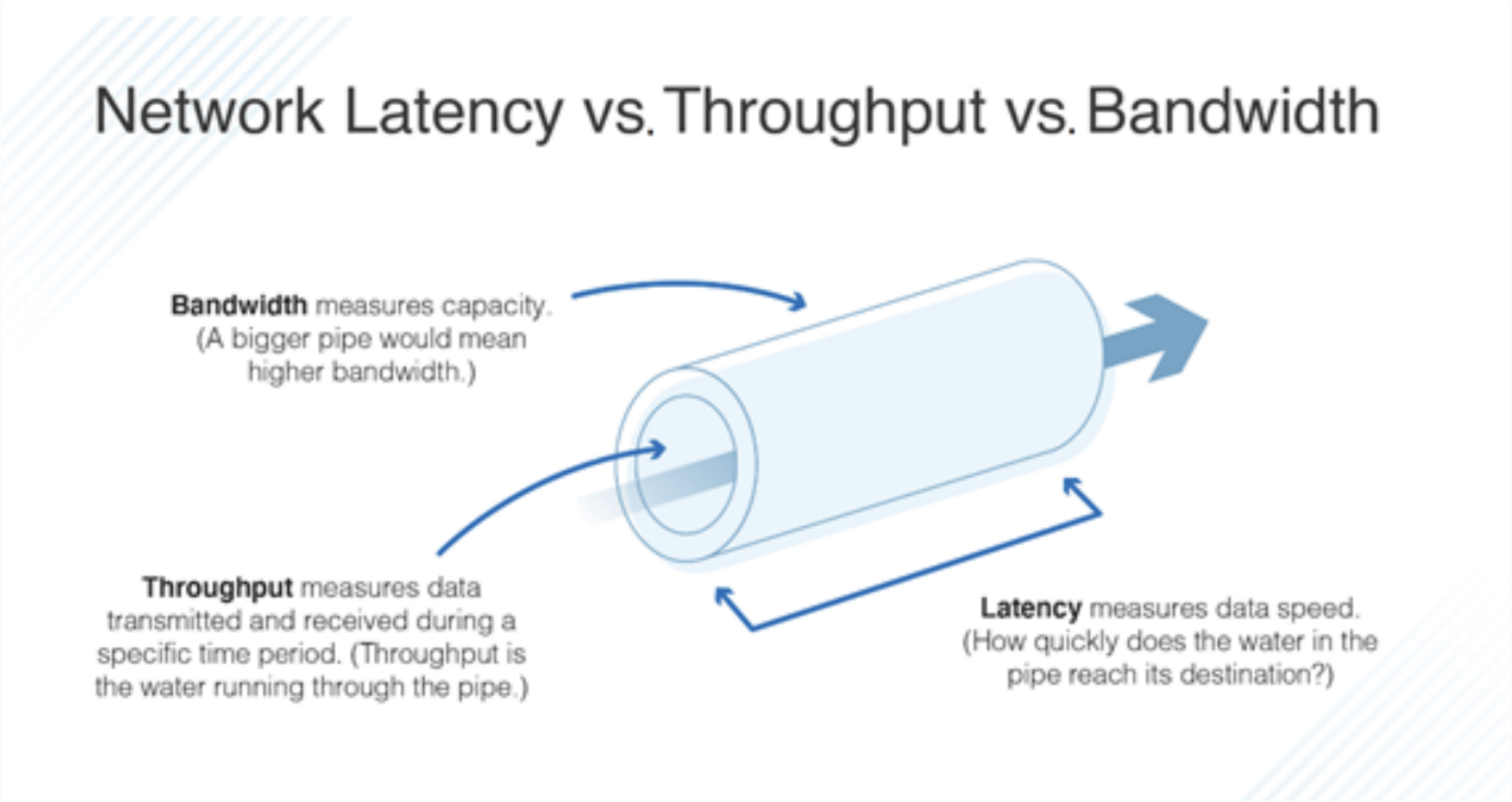
- Fraud detection: build a model to detect credit card fraud using thousands of features and billions of transactions
- Bioinformatics: easily manipulate genomic data from thousands of people to detect genetic associations with disease
- Finance: estimate financial risk through simulations of portfolios that include millions of instruments
- Monitoring of IoT applications: assess agricultural land use and crop yield for improved policymaking by periodically processing millions of satellite images
- Recommender systems: intelligently recommend millions of products to millions of users
- Web analytics, Social media analytics

SCALABLE MACHINE LEARNING

SCALABILITY

Ability of a system to maintain performance under increased load by adding more resources.

- Load: amount of existing data, rate of incoming data and required quality of service (e.g. number of queries per second)
- Linearly scalable: performance is maintained by adding resources in proportion to the increased load
- A nonlinearly scalable system is typically not very useful
- NB: a scalable system does not necessarily address latency and throughput at the same time.
- Kind of parallelism determined by **granularity** or grain size (i.e. the amount of work per process): this is measured by the ratio computational time over communication time.

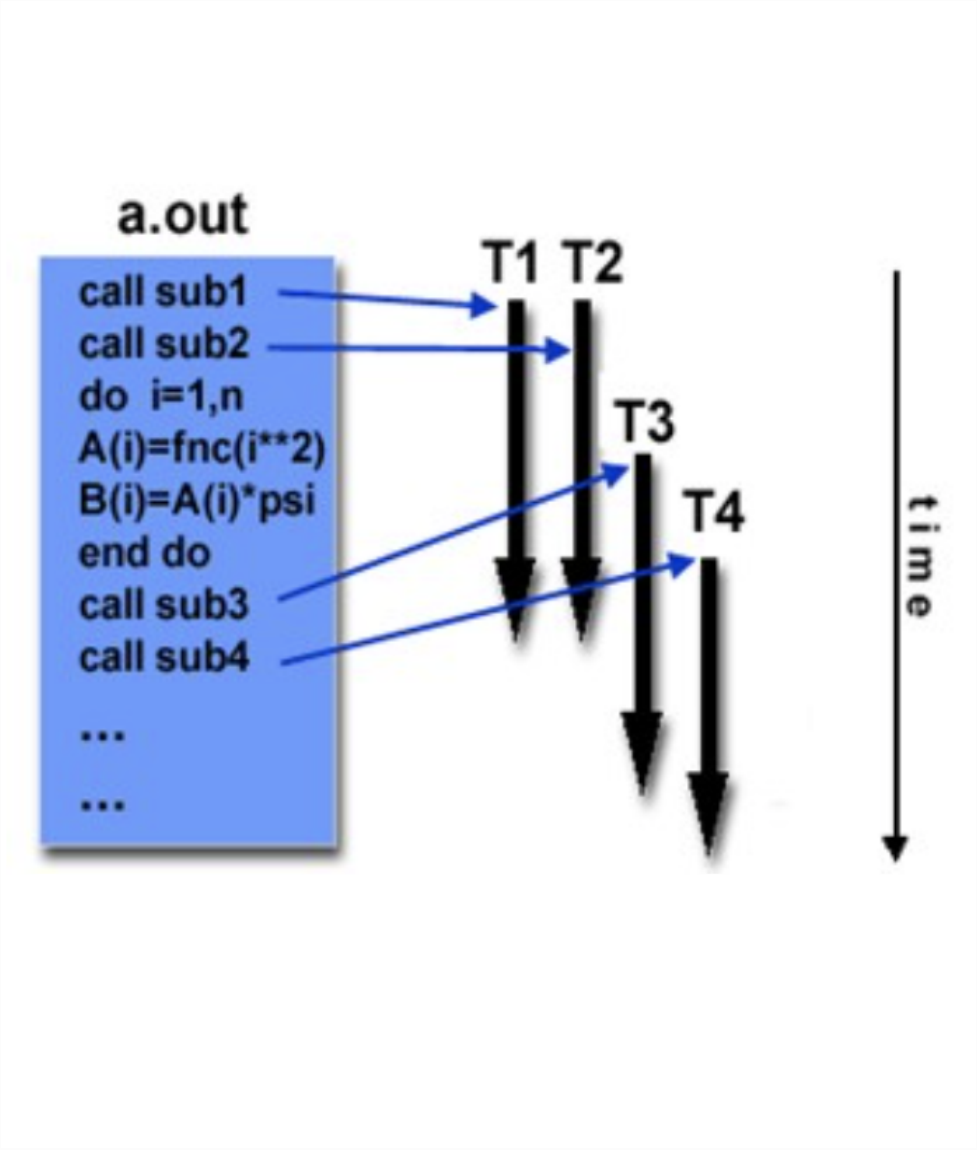


- Throughput: The number of tasks or operations a system can handle per unit of time as resources are added.
- Latency: The time taken to process a single task or query

FINE GRAINED PARALLELISM

- **Task parallelism**: segmentation into **large** number of **small** tasks, which may be executed concurrently
- Small amount of computational work between communication events
- Algorithm as a dataflow of elementary tasks (functional decomposition).
- Significant expertise required.
- High number of processors but amount of work per task is low
- Use of supercomputer, special-purpose parallel computers with many processors, shared memory and specialized hardware (GPUs and FPGAs)
- Ad hoc compilers, conventional operating system
- **Scale-up (vertically)**: scalability is obtained by adding more resources (e.g. faster PUs or more memory)
- It typically **reduces latency**.

FINE GRAINED PARALLELISM



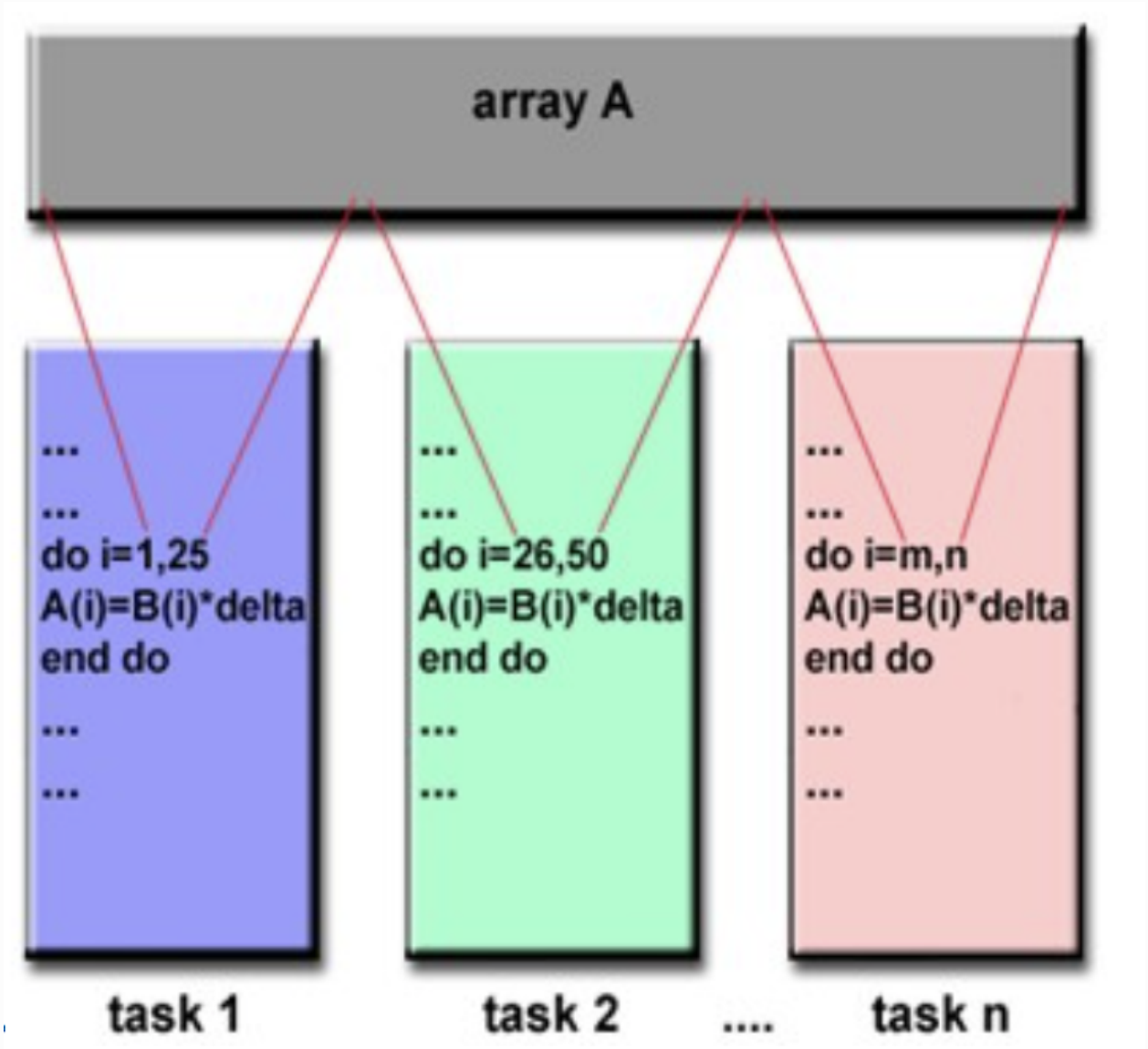
Instruction-level or loop-level parallelism

COARSE-GRAINED PARALLELISM

A program is split into small number of large (high granularity) tasks

- Large amount of computational work between communication and synchronization
- **Data parallelism**: concurrent calculations over portions of data
- Adequate for high volume tasks. Move code where data is
- Use of clusters, large collection of commodity hardware, including conventional processors ("compute nodes") connected by Ethernet cables or inexpensive switches.
- distributed file system (DFS), providing replication of data or redundancy against failure (Google File System (GFS), Hadoop Distributed File System (HDFS) and Cloud Store).
- Use of conventional languages extended by API.
- **Scale-out (horizontally)**: scalability is obtained by adding more commodity nodes.
- Increased overhead and communication costs associated with having more machines, risk of imbalance
- It typically **increases throughput**.

COARSE GRAINED PARALLELISM



BIG DATA VS HPC

HPC	Big data
Large computational load	large, complex datasets
High speed	Massive storage
Fine-grained (GPU)	Coarse-grained
Low latency	High-throughput
Data in local memory	Distributed data
Data in memory	Data in disk
Simulation, modeling	Analytics, Machine learning, AI
No fault tolerant	Fault tolerant
MPI	Hadoop, Spark
Scale up	Scale-out

SCALABLE MACHINE LEARNING

Multiple reasons to parallelize learning algorithms

- large volume of data instances and data resident on distributed file systems (vertical big data)
- high input dimensionality (horizontal big data)
- complexity of model parametric identification
- model selection and hyperparameters calibration
- resampling and averaging approaches
- performance requirements (e.g. latency and throughput)

EMBARRASSINGLY PARALLEL TASKS

- No dependencies (for instance, there is no need to communicate data among them in a message-passing setting).
- No task-to-task communication to complete its parts of the computation, so their computing/communication ratio is infinity.
- Example: bagging or Monte Carlo simulation tasks (no synchronization with their peer tasks from start to finish).
- High computation/communication ratio (time a task spends actually computing divided by the time it spends communicating)

HIGH VOLUMES

If data stored in a rectangular matrix where the rows are instances and the columns are the features.

- Many instances: data parallelism by partitioning the matrix rowwise into subsets of instances that are then processed independently (e.g., parameters update)
- Many features: data parallelism by splitting it columnwise for algorithms that can decouple the computation across features (e.g., decision trees).

Drawbacks: assumptions of i.i.d samples, independent features

VOLUME: VERTICAL BIG DATA

Domains where streams of data are measured and collected sequentially:

- internet
- finance (event is a transaction)
- business transactions (e.g. credit card transaction)
- sensors (Internet of Things, robots, cameras)

VOLUME: HORIZONTAL BIG DATA

Domains where instances are described by a very large number of feature (or attributes):

- Natural language processing: a feature per word
- Bioinformatics: a feature per genetic/genomic feature (e.g. variant, gene expression, methylation)
- Images: a feature per pixel

Data are sometimes sparse yet not always

SCALABLE PARAMETRIC IDENTIFICATION

Parametric identification is an example of multivariate optimization problem.

- In nonlinear models use of long and time intensive training procedures (e.g. multi-layer (deep) neural networks)
- Fine-grained parallel implementation of parametric identification procedures (e.g. backpropagation in neural networks) in multiprocessor or custom processor solutions (e.g. GPU)
- Coarse-grained decomposition at the sample level (e.g. in gradient descent algorithms)
- Ad-hoc solutions for tasks like image or video processing

SCALABLE MODEL ASSESSMENT AND VALIDATION

Examples of embarrassing parallel tasks in machine learning are

- Grid search (parameter sweeping) on large hyperparameter spaces
- Validation procedures like cross-validation, leave-on-out, bootstraping
- Bagging, averaging
- Univariate feature ranking
- Model racing and selection
- Subsampling or Monte Carlo strategies
- Multiple statistical significance testing (e.g. parallel p-value computing)
- Estimation of learners' variance

OTHER SCALABLE TASKS

- Similarity search, hashing
- Data-stream processing, online learning
- Search engine
- Graph mining

METRICS OF PARALLELIZATION PERFORMANCE

- Classical analysis of algorithms complexity is based on O-notation to quantify computational costs.
- This is hardly usable in machine learning, since the execution time (e.g. related to termination conditions) is often dependent on data distribution which is not known a priori
- Key metrics used for analyzing computational performance of parallel algorithms are
  1. Speedup
  2. Scaleup
  3. Sizeup
- Don't forget that these measures should be also be assessed
  - in paired settings (e.g. same dataset)
  - in association with generalization accuracy

SPEED-UP

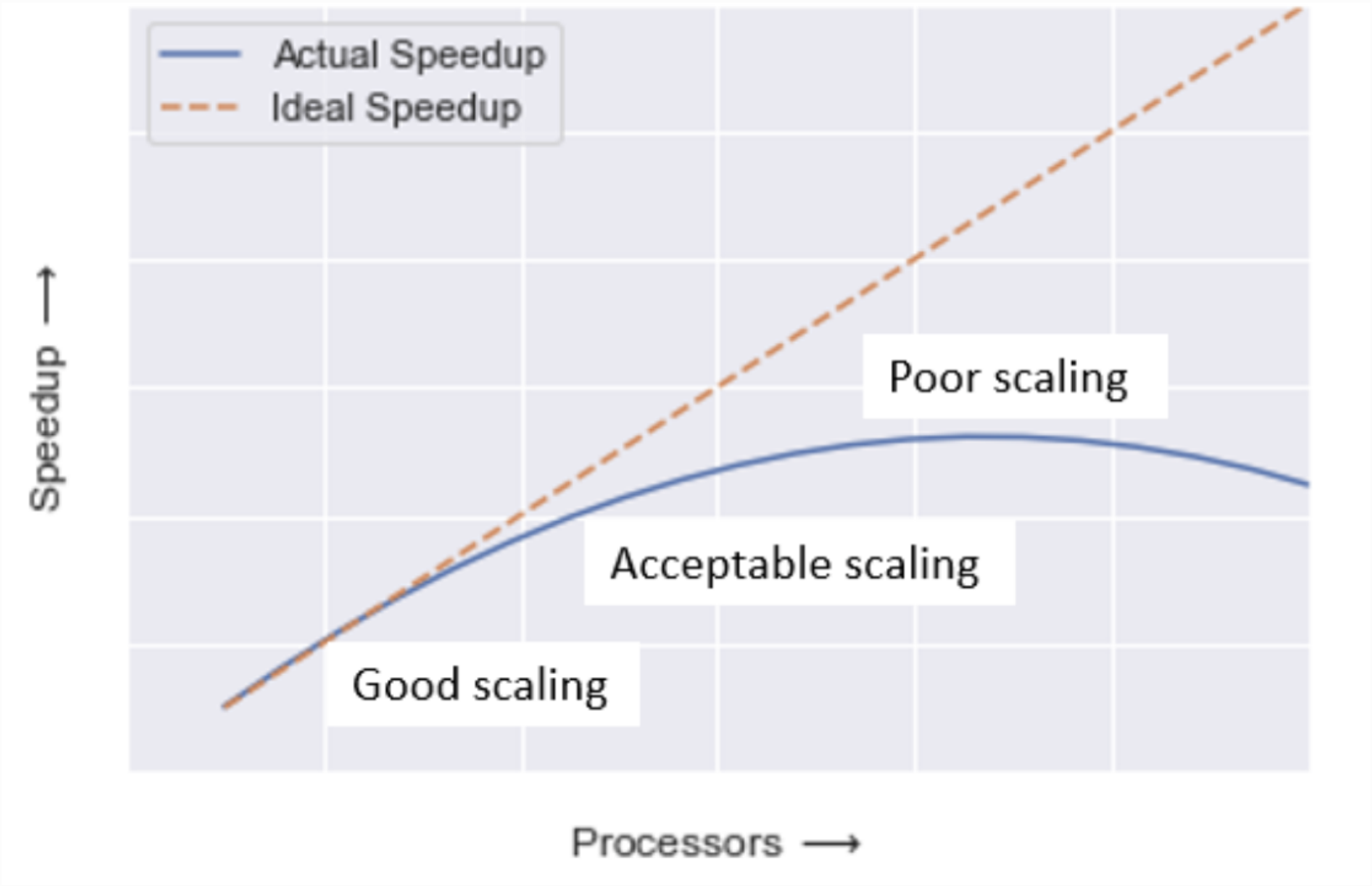
Let $T(N, p)$ the execution time of an algorithm where N is the size of the input data and p is the number of processors.

- Speed-up refers to how much a parallel algorithm is faster than a corresponding sequential algorithm.
- Speed-up analysis holds the input size and equal to N, and grows the number p of workers. It is defined by the following formula:

$$\text{Speed-up}(p) = \frac{T(N, 1)}{T(N, p)}$$

  where $T(N, p)$ is the execution time with p workers.
- Linear speed-up or ideal speedup is obtained when the speed-up is a linear function of p (e.g. by doubling the number of processors we double the speed).
- Linear speedup is difficult to achieve because of communication costs.

SPEED-UP


Actual Speedup
Ideal Speedup
Speedup ⟶
Poor scaling
Acceptable scaling
Good scaling
Processors ⟶

SIZE-UP

- Size-up analysis holds the number of workers in the system constant and equal to p, and grows the size N of the datasets by the factor m.
- Size-up measures how much longer it takes on a given system, when the dataset size is m-times larger than the original dataset.
- It is defined by the following formula:

$$\text{Size-up}(m) = \frac{T(mN, p)}{T(N, p)}$$

where $T(mN, p)$ is the execution time of the algorithm for processing a data set of size mN

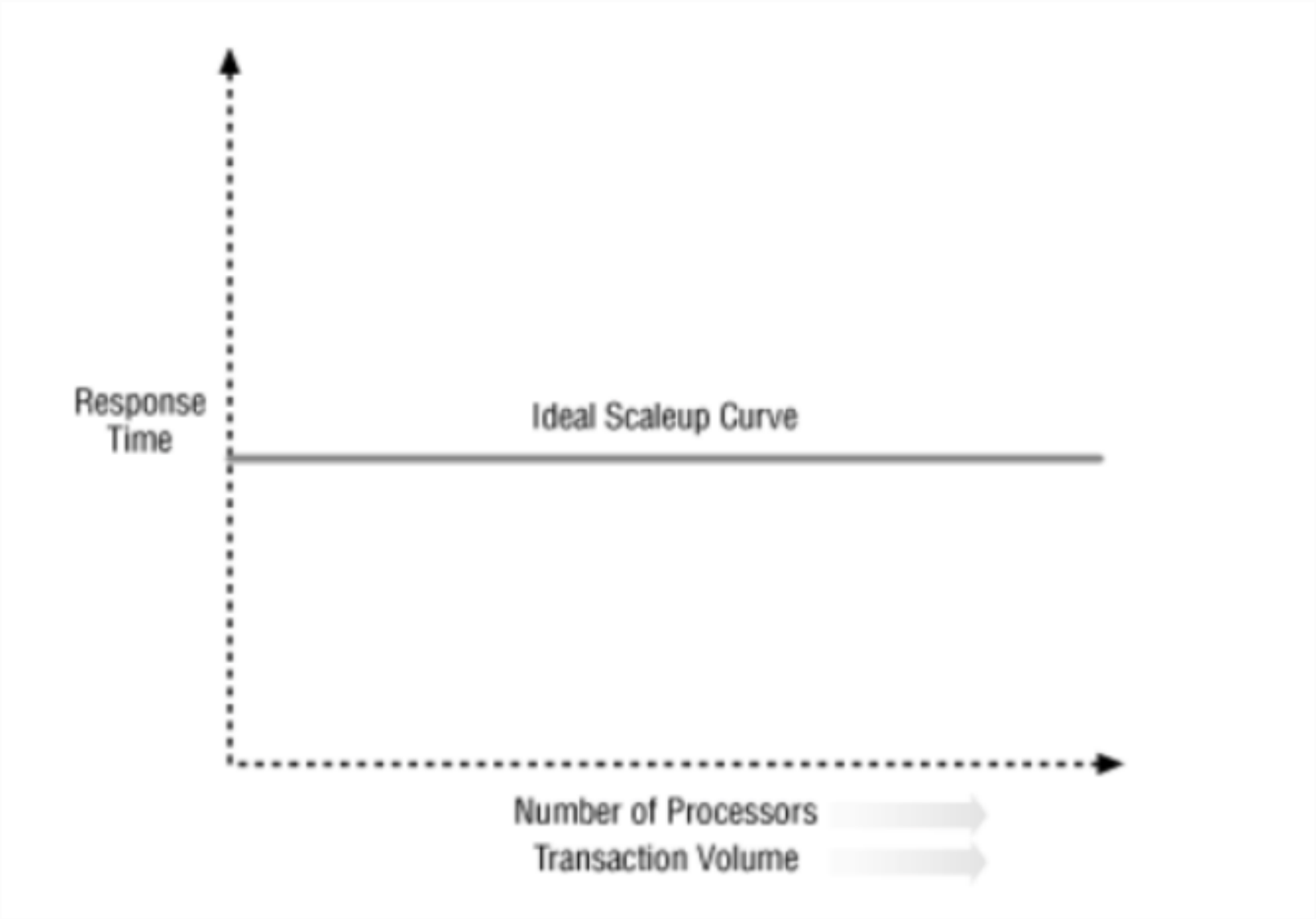
SCALE-UP

- Scale-up evaluates the ability of the algorithm to deal with the growth of both the system and the dataset size.
- Scale-up is defined as the ability of a m-times larger system to perform a m-times larger job in the same run-time as the original system.
- It is defined by the following formula:

$$\text{Scale-up}(m) = \frac{T(N, p)}{T(mN, mp)}$$

- Scale-up experiments are performed by increasing the size of the datasets in direct proportion to the number of cores in the system.
- The ideal scale-up curve is constant.

SCALE-UP


Response
Time
Ideal Scaleup Curve
Number of Processors
Transaction Volume

Raja Appuswamy, Christos Gkantsidis, Dushyanth Narayanan, Orion Hodson, and Ant Rowstron.
Nobody ever got fired for buying a cluster.
Technical report, January 2013.

B. Baesens.
Analytics in a Big Data World: the essential guide to data science and its applications.
Wiley, 2014.

G. Bontempi.
Statistical Foundations of Machine Learning: the handbook.
2012.
Handbook of INFOF422 course.