# **CODES** for **SPEED:**

## the **CLEAR** advantage
## (**C**omputing, **LE**arning, **A**ccess, and **R**ecovery)

Kannan Ramchandran

University of California, Berkeley

BLISS

Berkeley Laboratory for Information and System Sciences

# Shannon's legacy

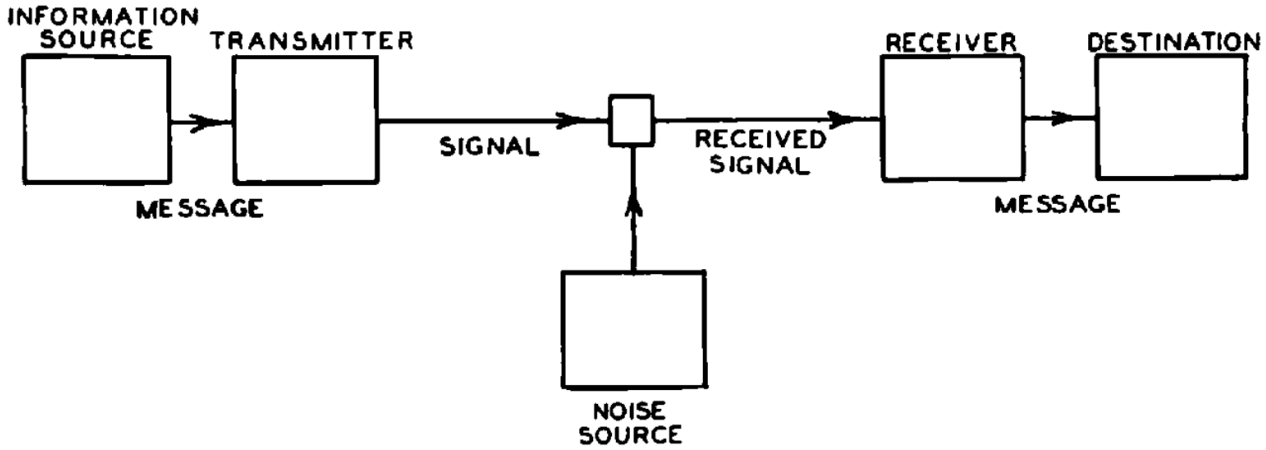
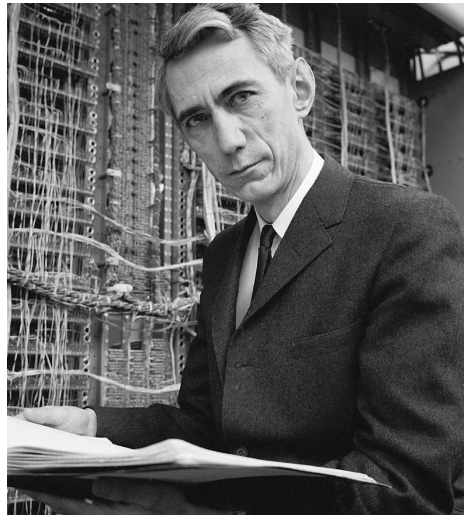



Fig. 1—Schematic diagram of a general communication system.



**SOURCE CODING**

Can represent a source using ***bits > entropy***

**CHANNEL CODING**

Can transmit with arbitrarily small error probability at ***rates < channel capacity***

**1948*

# Impact of coding theory

**Source coding**

(1952) Huffman

(1967) Tunstall

(1976) Arithmetic

(1977) Lempel-Ziv

**Channel coding**

Hamming (1947)

Reed-Solomon (1960)

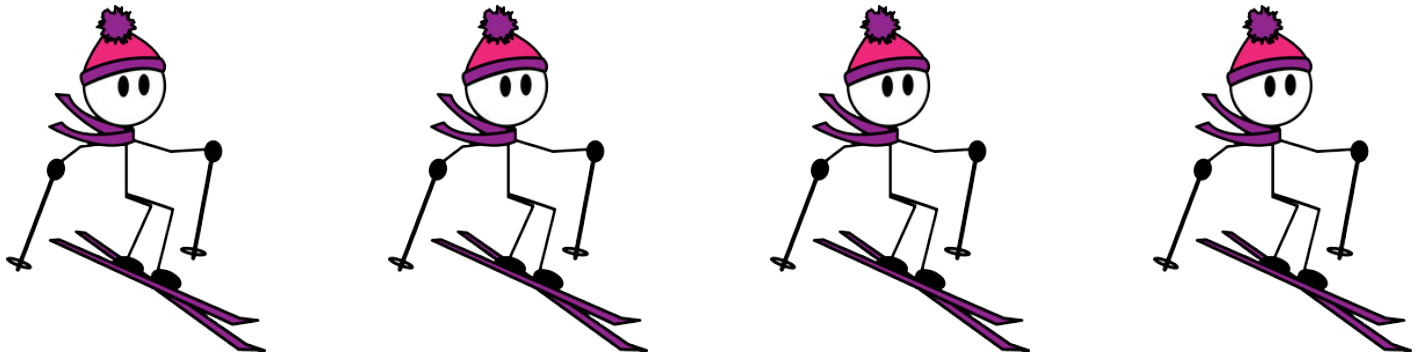
LDPC (1963)

Turbo (1990)

Polar (2009)

# Primer on error correcting codes

- **3 data chunks** $X_1, X_2, X_3$ to deliver
- **4 skiers**, each can carry 1 chunk-size data
- You know **1 of them will get delayed** but don't know which



$X_1$ $X_2$ $X_3$ $X_4 = X_1 + X_2 + X_3$

# Primer on error correcting codes

- **3 data chunks** $X_1, X_2, X_3$ to deliver
- **4 skiers**, each can carry 1 chunk-size data
- You know **1 of them will get delayed** but don't know which

$X_1$ $X_2$ $X_3$ $X_4 = X_1 + X_2 + X_3$

Recover $X_2 = X_4 - X_1 - X_3$

# Error correcting codes

Used extensively in **communications** and **storage**

**Low density parity check (LDPC) codes**

- Wi-Fi 802.11
- DVB-S2
- 10GBase-T Ethernet
- WiMAX

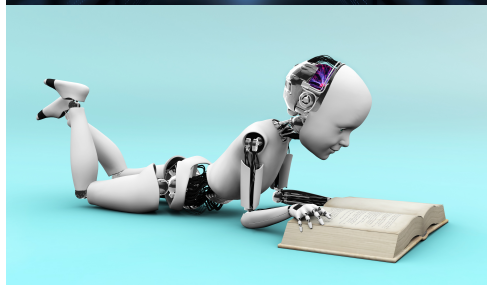
**Reed-Solomon (RS) codes**

- Cloud storage
- QR codes
- CD, DVD, Blu-ray
- DSL
- RAID 6
- WiMAX

# Outward view of codes

# The **CLEAR** advantage

**Computing**



**LEarning**

**Access**

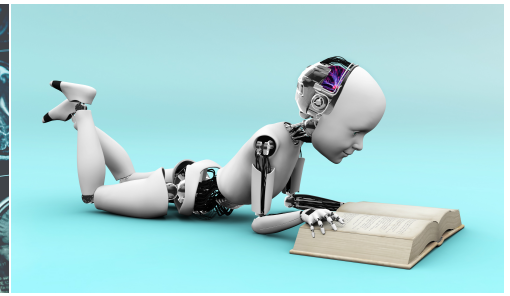
**Recovery**

**CODES for SPEED**

# Outline

**Access**

**Recovery**



**LEarning**

**Computing**

Alex
Dimakis

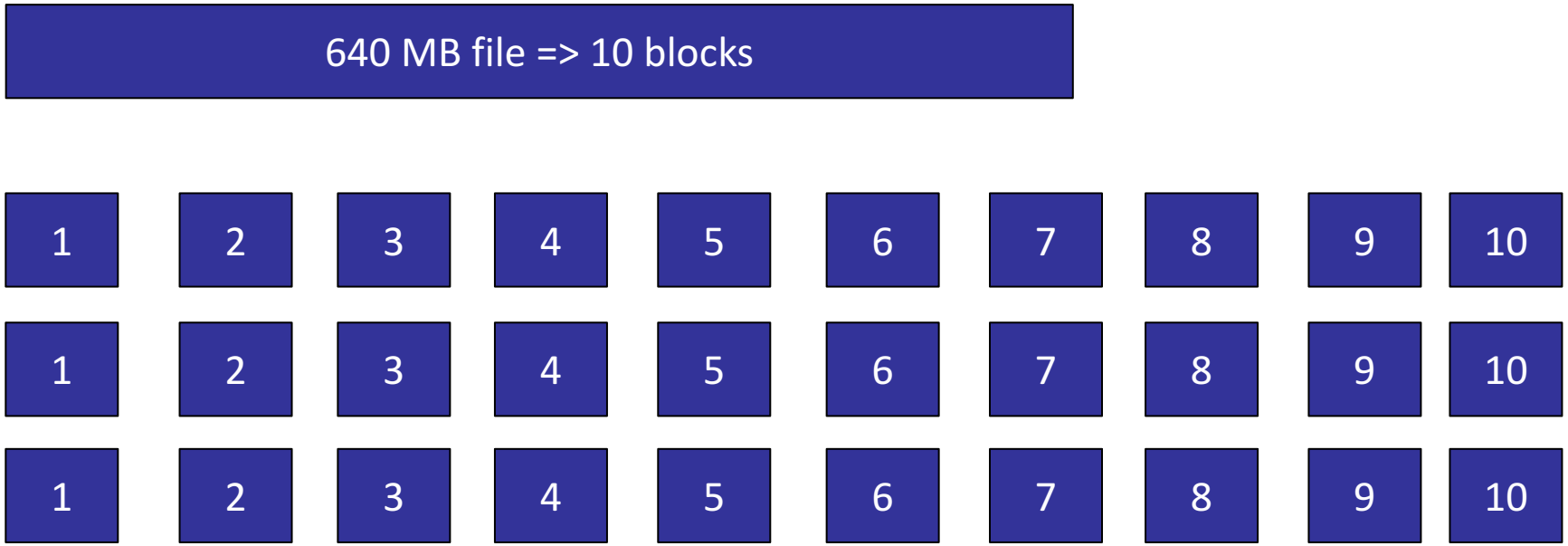
Rashmi
Vinayak

Nihar Shah

# Chapter 1

## Speeding up data access in distributed storage systems

# Hadoop Distributed File System (HDFS)
# Default 3x replication




## FB Warehouse Cluster

- Multiple tens of PBs and growing
- Multiple thousands of nodes

**Storage efficiency → main driver for cost**

# HDFS-RAID: (14, 10) MDS code

640 MB file => 10 blocks

1 2 3 4 5 6 7 8 9 10

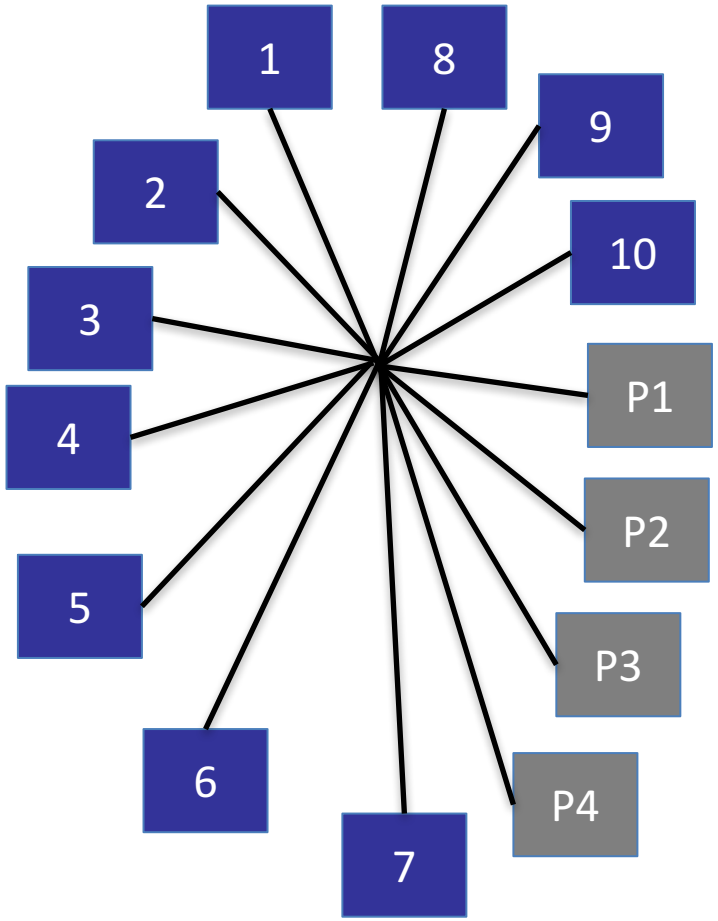
P1 P2 P3 P4

FB Warehouse Cluster

- Reduce storage overhead from x3 to x1.4 (53% reduction)
- For less frequently accessed data

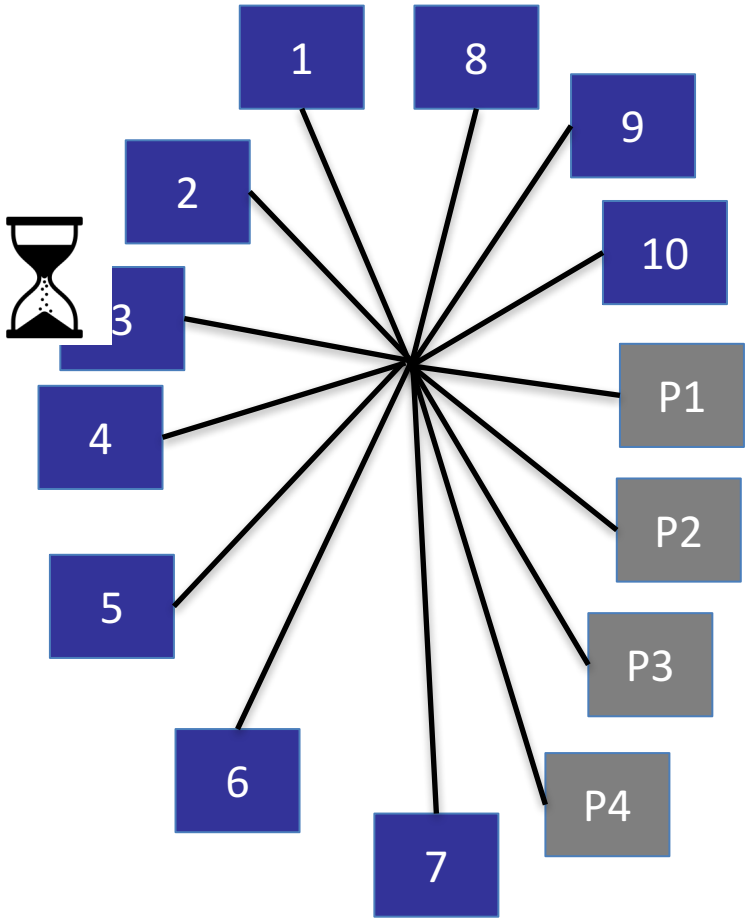
- What about the ***speed of access*** ? (especially in the event of unavailability)

# Codes are good but…




Good news: we can now tolerate 4 node failures.

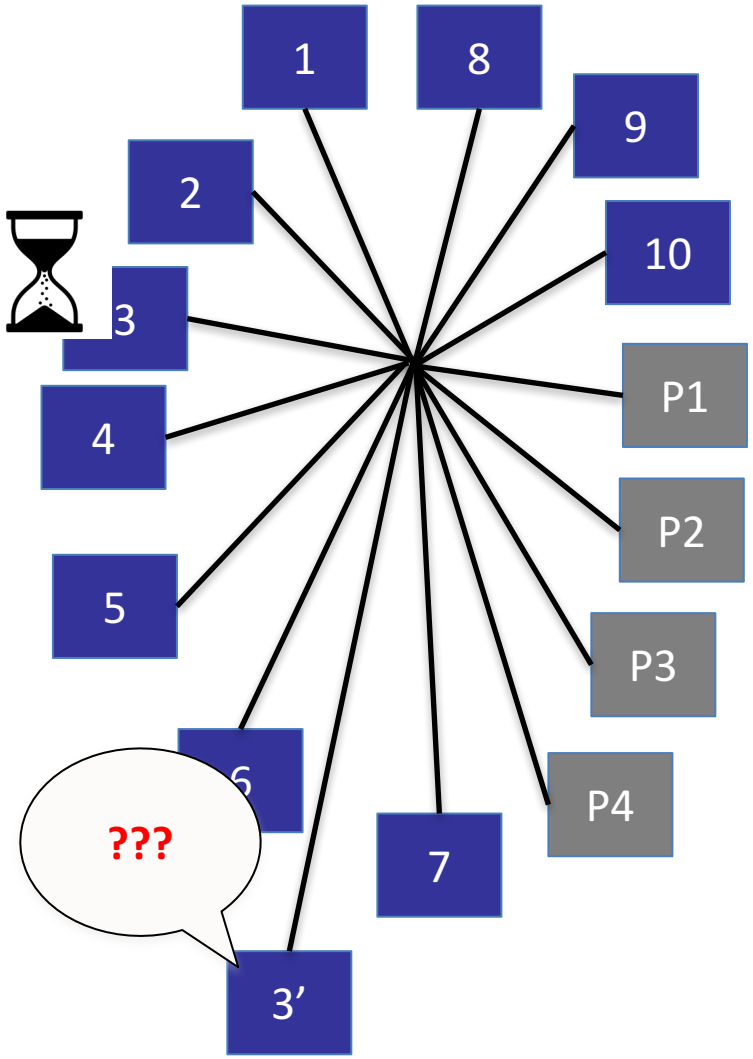
# Codes are good but…




Good news: we can now tolerate 4 node failures.

User wants 3 but the server is unavailable

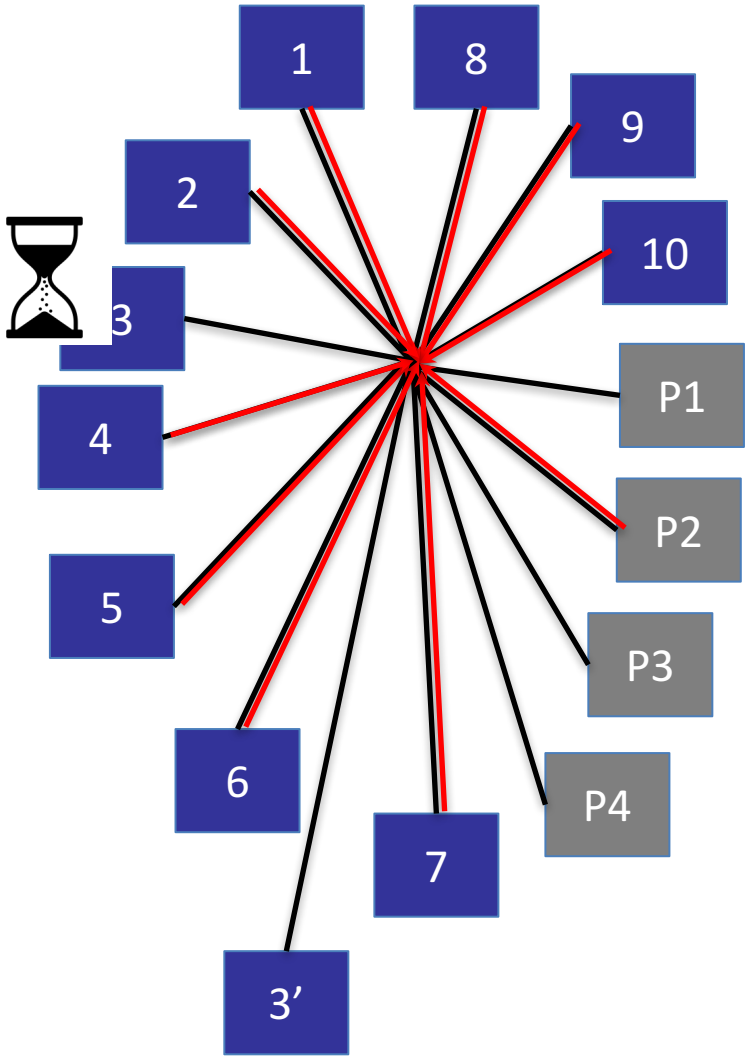
# Codes are good but…




Good news: we can now tolerate 4 node failures.

User wants 3 but the server is unavailable

# Codes are good but…

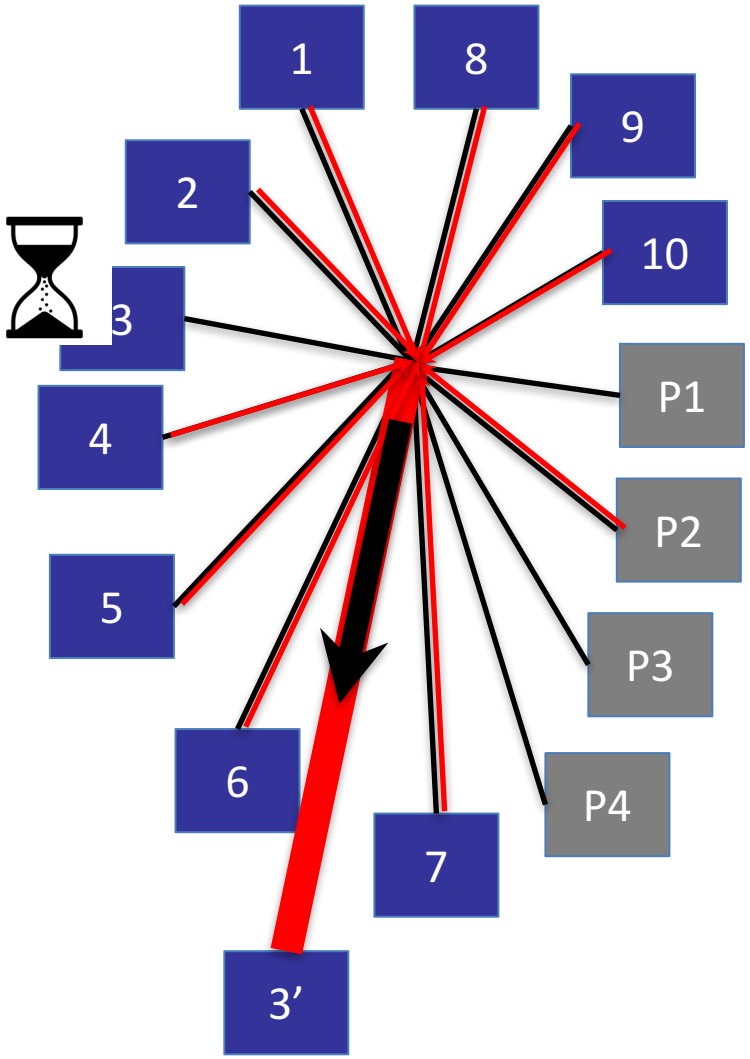



Good news: we can now tolerate 4 node failures.

User wants 3 but the server is unavailable

Read from any 10 nodes, send all data to 3' to repair the lost block.

# Codes are good but…




Good news: we can now tolerate 4 node failures.

User wants 3 but the server is unavailable

Read from any 10 nodes, send all data to 3' to repair the lost block.

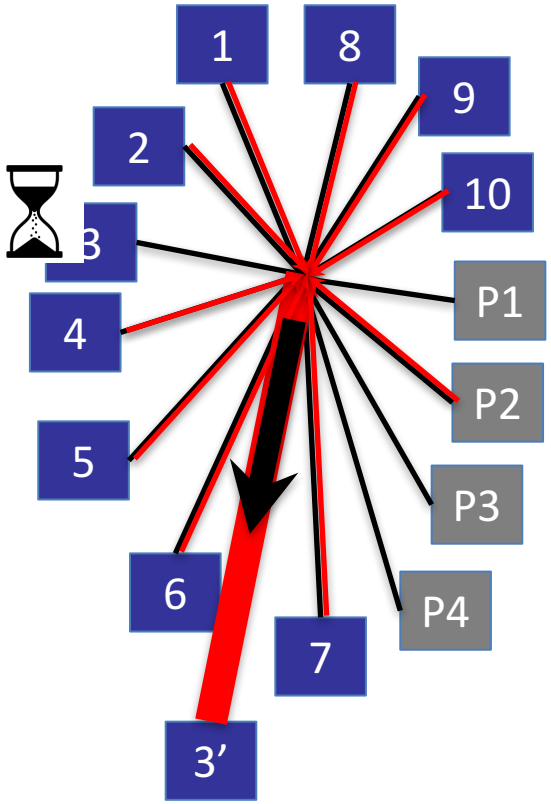
Bad news:

- `Degraded' read is slow
- High network traffic
- High disk read (10x more than the lost information)

# Best of both worlds possible ?

Can we have

- **Storage efficiency** of codes
- **BW efficiency (speed)** of replication



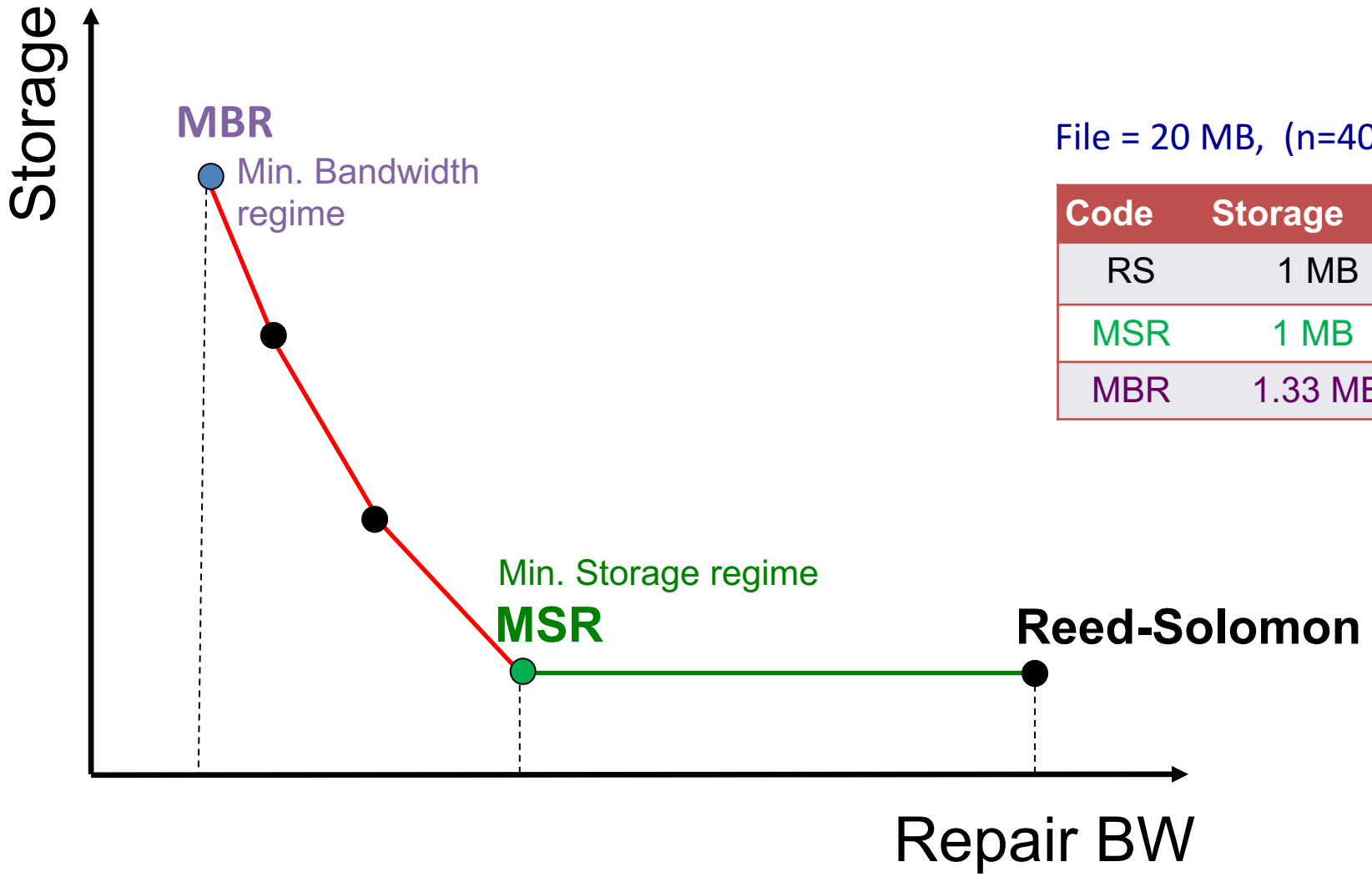

Almost…

… there exists (an optimal) tradeoff

**Regenerating Codes**

*Dimakis, Godfrey, Wainwright & R, 2010*

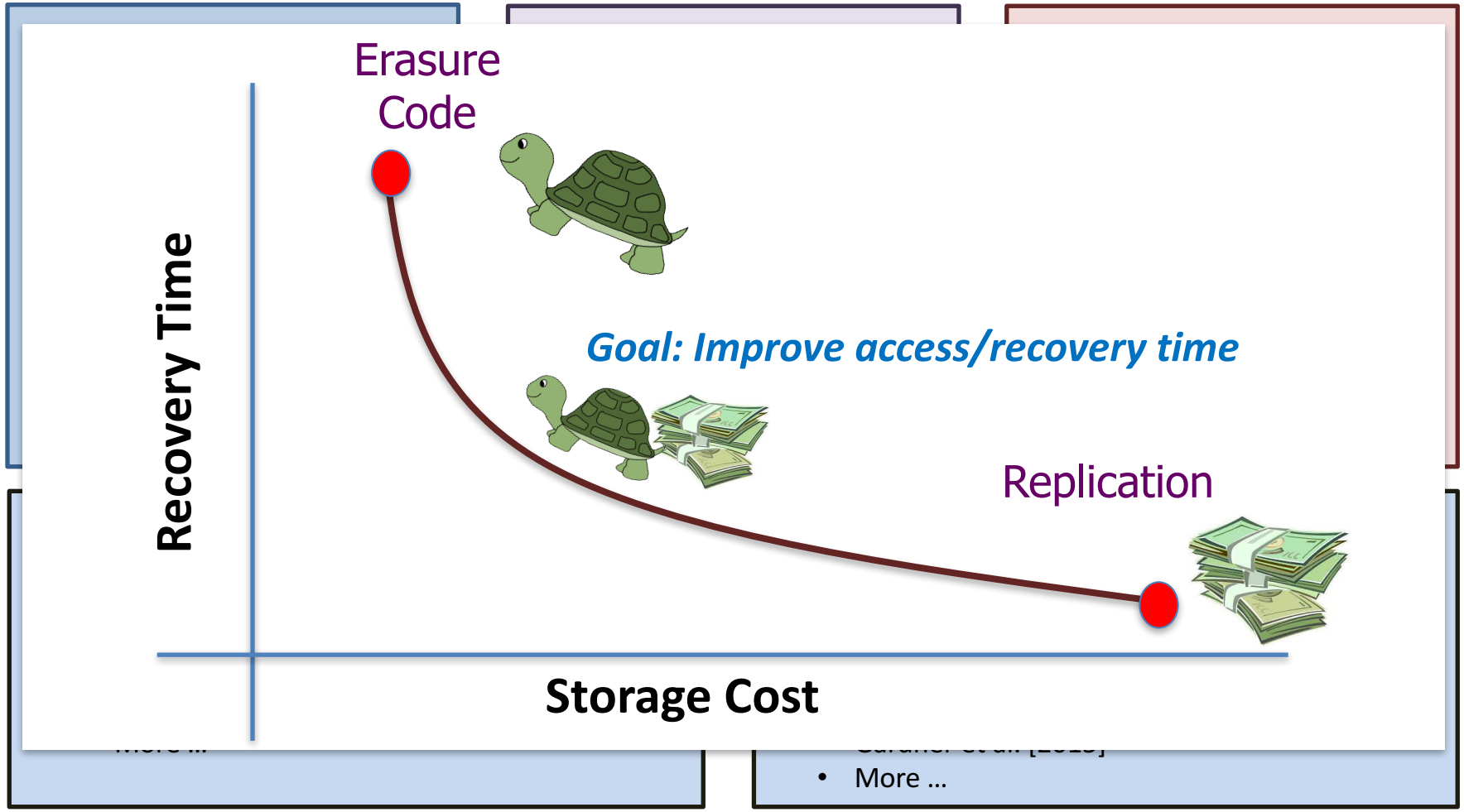
# Storage-Bandwidth tradeoff




File = 20 MB, (n=40, k=20) code

| Code | Storage | Bandwidth | Savings |
|---|---|---|---|
| RS | 1 MB | 20 MB | |
| MSR | 1 MB | 2 MB | 10x |
| MBR | 1.33 MB | 1.33 MB | 15x |

# Coding for Storage: Active Area of Research




- More …

# Industry impact

**Piggyback codes & Hitchhiker**

[Rashmi et al. ISIT 13, IT-Tran 17]
[Rashmi et al. SIGCOMM 14]

**Butterfly codes**

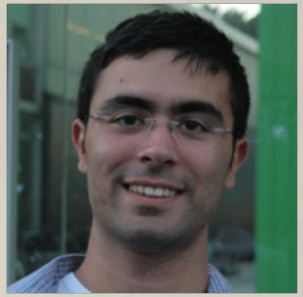
[En Gad et al. ISIT 13]
[Pamies-Juarez et al. FAST 16]

**Ye-Barg codes & Clay codes (IT-Tran 17, FAST 18)**

[Ye and Barg IT-Tran 17]
[Vajha et al. FAST 18]

Sameer Pawar

Simon Li

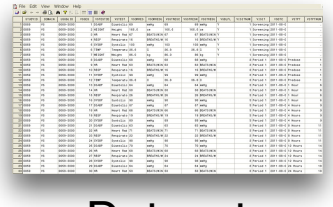


Orhan Ocal

# Chapter 2

## Speeding up learning and recovery of sparse signals

# Motivation

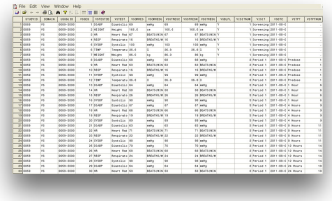


Dataset

# Motivation

- Given training data points $(x, y)$, our goal is to learn

$(x$ : feature, $y$ : label$)$

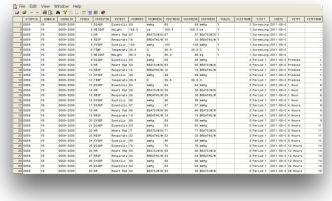


Dataset

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$

$(x$ : feature, $y$ : label$)$

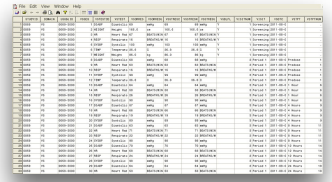


Dataset

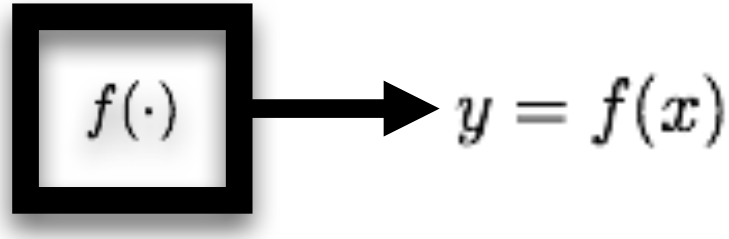
# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$

$(x$ : feature, $y$ : label$)$



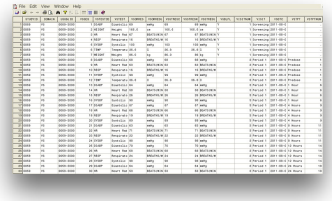
Dataset



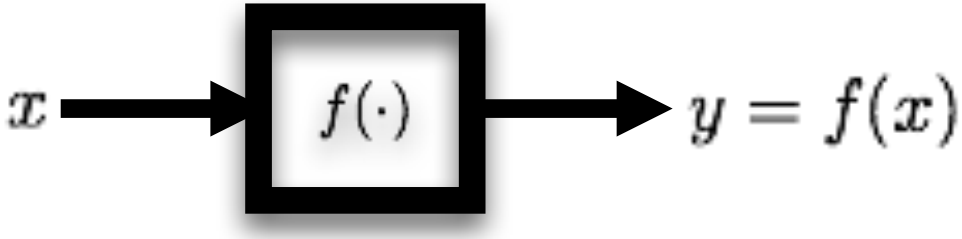
# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label$)$



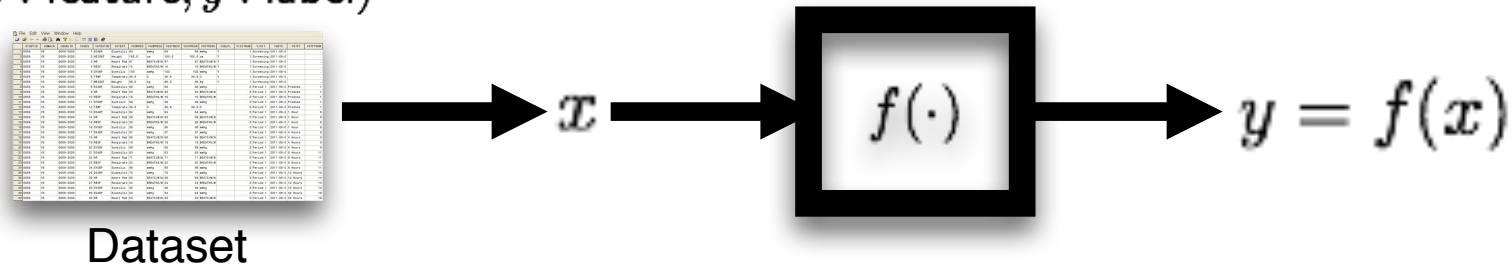
Dataset



# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

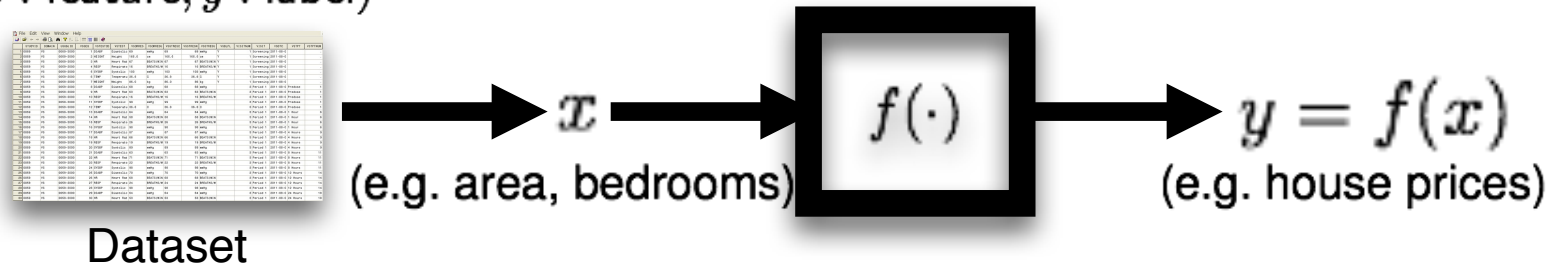
$(x$ : feature, $y$ : label$)$



# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

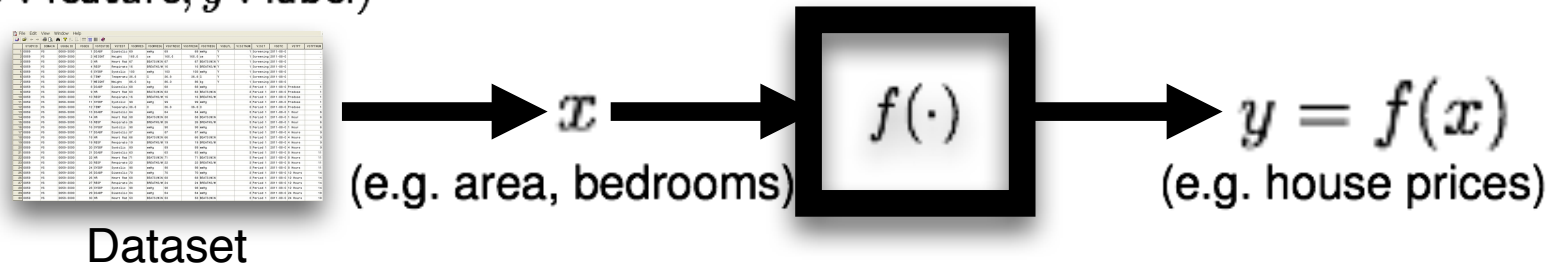
$(x$ : feature, $y$ : label$)$



# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label$)$

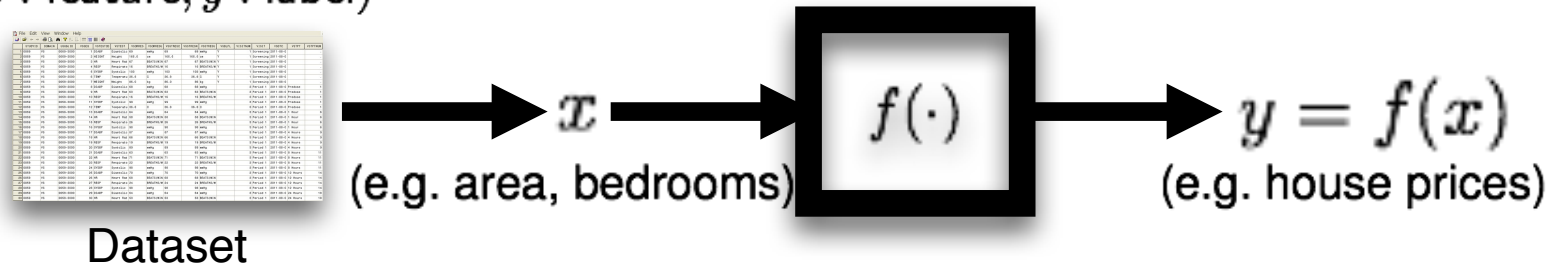



- Questions of interest
  - Sample complexity: how many data points do we need?

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label$)$

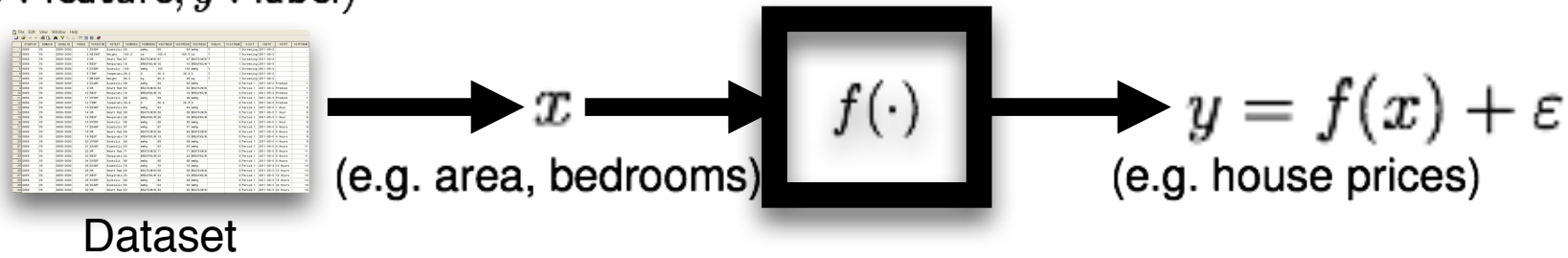



- Questions of interest
  - Sample complexity: how many data points do we need?
  - Computational complexity: how much time does it take?

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label$)$

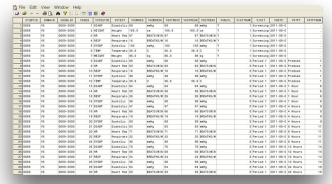



- Questions of interest
  - Sample complexity: how many data points do we need?
  - Computational complexity: how much time does it take?
  - Robustness: how accurate and stable is it?

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label$)$



Dataset

$x$

(e.g. area, bedrooms)

$f(\cdot)$

$y = f(x) + \varepsilon$

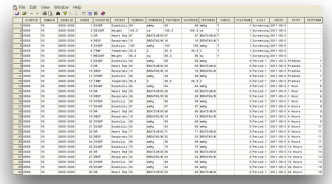
(e.g. house prices)

e.g. $f(x_1, x_2) = a_1 x_1 + a_2 x_2 + b$

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label$)$



Dataset

$x$ (e.g. area, bedrooms) $\rightarrow$ $f(\cdot)$ $\rightarrow$ $y = f(x) + \varepsilon$ (e.g. house prices)

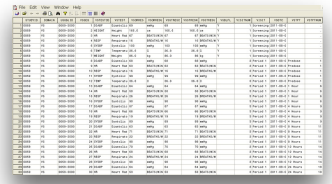
e.g. $f(x_1, x_2) = a_1 x_1 + a_2 x_2 + b$

Problem Dimension $N = 3$

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label$)$



Dataset

$x$ (e.g. area, bedrooms) → $f(\cdot)$ → $y = f(x) + \varepsilon$ (e.g. house prices)

e.g. $f(x_1, x_2) = a_1 x_1 + a_2 x_2 + b$

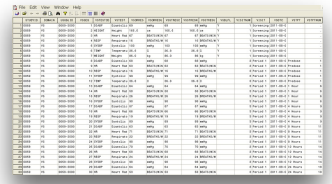
**EASY!**

Problem Dimension $N = 3$

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label$)$



Dataset

$x$ (e.g. area, bedrooms) $\rightarrow$ $f(\cdot)$ $\rightarrow$ $y = f(x) + \varepsilon$ (e.g. house prices)

e.g. $f(x_1, x_2) = a_1 x_1 + a_2 x_2 + b$

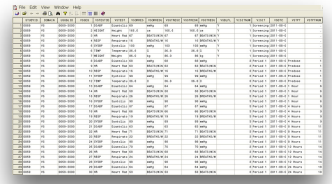
**However…**

Problem Dimension $N = 3$

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label$)$



Dataset

$$\xrightarrow{\text{(e.g. area, bedrooms)}} \; x \;\longrightarrow\; \boxed{f(\cdot)} \;\longrightarrow\; y = f(x) + \varepsilon$$

(e.g. house prices)

e.g. $f(x_1, x_2) = a_1 x_1 + a_2 x_2 + b$

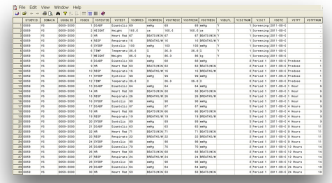
**in reality…**

Problem Dimension $N = 3$

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label)



Dataset

$$x \longrightarrow \boxed{f(\cdot)} \longrightarrow y = f(x) + \varepsilon$$

e.g. $f(x) =$

$$\frac{D}{Dt}\overline{w'^i w'^j} + \overline{w'^l w'^\alpha}\nabla_\alpha \bar{u}^j + \overline{w'^j w'^\alpha}\nabla_\alpha \bar{u}^i - \alpha\left(g^{i\alpha}\overline{w'^j\frac{T'}{\bar{T}}} + g^{j\alpha}\overline{w'^i\frac{T'}{\bar{T}}}\right)\left(\nabla_\alpha\bar{\Phi} + \frac{D\bar{u}_\alpha}{Dt}\right)$$

$$+\frac{1}{\bar{\rho}}\nabla_\alpha[\bar{\rho}\overline{u'^\alpha w'^i w'^j} + \overline{(g^{i\alpha}w'^j + g^{j\alpha}w'^i)P'} - \overline{w'^i\sigma^{j\alpha}(u')} - \overline{w'^j\sigma^{i\alpha}(u')}]$$

$$+\frac{1}{\bar{\rho}}\overline{w'^i w'^j\nabla_\alpha(\bar{\rho}u'^\alpha)} - \overline{P'(g^{i\alpha}\nabla_\alpha w'^j + g^{j\alpha}\nabla_\alpha w'^i)} = -\frac{1}{\bar{\rho}}[\overline{\sigma^{i\alpha}(u')\nabla_\alpha w'^j} + \overline{\sigma^{j\alpha}(u')\nabla_\alpha w'^i}] = -\epsilon_2^{ij}\,,$$

$$(1+e_4)\frac{D}{Dt}\overline{\left(\frac{T'}{\bar{T}}\right)^2} - 2f(t)\overline{\left(\frac{T'}{\bar{T}}\right)^2} - 2\overline{w'^\alpha\frac{T'}{\bar{T}}}D_\alpha + \frac{1}{(1+e_4)\bar{\rho}C_p^2}\nabla_\alpha\left[(1+e_4)^2C_p^2\bar{\rho}\overline{w'^\alpha\left(\frac{T'}{\bar{T}}\right)^2}\right] + \frac{1+e_4}{\bar{\rho}}\overline{\left(\frac{T'}{\bar{T}}\right)^2\nabla_\alpha(\rho u'^\alpha)}$$

$$+\frac{2}{\bar{\rho}\bar{T}C_p}\overline{\frac{T'}{\bar{T}}\left[P'\nabla_\alpha w'^\alpha - \nabla_\alpha(P'_g w'^\alpha) - \frac{DP'_g}{Dt}\right]} = \frac{2}{\bar{\rho}\bar{T}C_p}\overline{\frac{T'}{\bar{T}}[\sigma^{\alpha\beta}(u')\nabla_\alpha u'_\beta - \nabla_\alpha F'^\alpha_r]} = -\epsilon_2\,,$$

$$(1+e_4)\left[\frac{D}{Dt}\overline{\left(w'^i\frac{T'}{\bar{T}}\right)} + \overline{w'^\alpha\frac{T'}{\bar{T}}}\nabla_\alpha\bar{u}^i - \alpha\overline{\left(\frac{T'}{\bar{T}}\right)^2}g^{i\alpha}\left(\nabla_\alpha\bar{\Phi} + \frac{D\bar{u}_\alpha}{Dt}\right)\right] - f(t)\overline{w'^i\frac{T'}{\bar{T}}} - \overline{w'^l w'^\alpha}D_\alpha$$

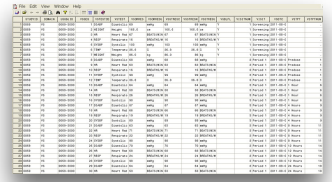
$$+\frac{1}{\bar{\rho}C_p}\nabla_\alpha\left[(1+e_4)C_p\bar{\rho}\overline{w'^i w'^\alpha\frac{T'}{\bar{T}}}\right] + \frac{1+e_4}{\bar{\rho}}\overline{w'^i\frac{T'}{\bar{T}}\nabla_\alpha(\rho u'^\alpha)} + \frac{1}{\bar{\rho}\bar{T}C_p}\overline{w'^i\left[P'\nabla_\alpha w'^\alpha - \nabla_\alpha(P'_g w'^\alpha) - \frac{DP'_g}{Dt}\right]}$$

**in reality…**

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label)



Dataset

$$x \longrightarrow \boxed{f(\cdot)} \longrightarrow y = f(x) + \varepsilon$$

e.g. $f(x) =$

$$\frac{D}{Dt}\overline{w'^i w'^j} + \overline{w'^i w'^\alpha}\nabla_\alpha \bar{u}^j + \overline{w'^j w'^\alpha}\nabla_\alpha \bar{u}^i - \alpha\left(g^{i\alpha}\overline{w'^j \frac{T'}{\bar{T}}} + g^{j\alpha}\overline{w'^i \frac{T'}{\bar{T}}}\right)\left(\nabla_\alpha \bar{\Phi} + \frac{D\bar{u}_\alpha}{Dt}\right)$$

$$+\frac{1}{\bar{\rho}}\nabla_\alpha[\bar{\rho}\overline{u'^\alpha w'^i w'^j} + \overline{(g^{i\alpha}w'^j + g^{j\alpha}w'^i)P'} - \overline{w'^i \sigma^{j\alpha}(u')} - \overline{w'^j \sigma^{i\alpha}(u')}]$$

$$+\frac{1}{\bar{\rho}}\overline{w'^i w'^j \nabla_\alpha(\bar{\rho}u'^\alpha)} - \overline{P'(g^{i\alpha}\nabla_\alpha w'^j + g^{j\alpha}\nabla_\alpha w'^i)} = -\frac{1}{\bar{\rho}}[\overline{\sigma^{i\alpha}(u')\nabla_\alpha w'^j} + \overline{\sigma^{j\alpha}(u')\nabla_\alpha w'^i}] = -\epsilon_2^{ij},$$

$$(1+e_4)\frac{D}{Dt}\overline{\left(\frac{T'}{\bar{T}}\right)^2} - 2f(t)\overline{\left(\frac{T'}{\bar{T}}\right)^2} - 2\overline{w'^\alpha \frac{T'}{\bar{T}}}D_\alpha + \frac{1}{(1+e_4)\bar{\rho}C_p^2}\nabla_\alpha\left[(1+e_4)^2 C_p^2 \bar{\rho}\overline{w'^\alpha\left(\frac{T'}{\bar{T}}\right)^2}\right] + \frac{1+e_4}{\bar{\rho}}\overline{\left(\frac{T'}{\bar{T}}\right)^2\nabla_\alpha(\rho u'^\alpha)}$$

$$+\frac{2}{\bar{\rho}\bar{T}C_p}\overline{\frac{T'}{\bar{T}}\left[P'\nabla_\alpha w'^\alpha - \nabla_\alpha(P'_g w'^\alpha) - \frac{DP'_g}{Dt}\right]} = \frac{2}{\bar{\rho}\bar{T}C_p}\overline{\frac{T'}{\bar{T}}[\sigma^{\alpha\beta}(u')\nabla_\alpha u'_\beta - \nabla_\alpha F'^\alpha_r]} = -\epsilon_2,$$

$$(1+e_4)\left[\frac{D}{Dt}\overline{\left(w'^i \frac{T'}{\bar{T}}\right)} + \overline{w'^\alpha \frac{T'}{\bar{T}}}\nabla_\alpha \bar{u}^i - \alpha\overline{\left(\frac{T'}{\bar{T}}\right)^2} g^{i\alpha}\left(\nabla_\alpha \bar{\Phi} + \frac{D\bar{u}_\alpha}{Dt}\right)\right] - f(t)\overline{w'^i \frac{T'}{\bar{T}}} - \overline{w'^i w'^\alpha}D_\alpha$$

$$+\frac{1}{\bar{\rho}C_p}\nabla_\alpha\left[(1+e_4)C_p\bar{\rho}\overline{w'^i w'^\alpha \frac{T'}{\bar{T}}}\right] + \frac{1+e_4}{\bar{\rho}}\overline{w'^i \frac{T'}{\bar{T}}\nabla_\alpha(\rho u'^\alpha)} + \frac{1}{\bar{\rho}\bar{T}C_p}\overline{w'^i\left[P'\nabla_\alpha w'^\alpha - \nabla_\alpha(P'_g w'^\alpha) - \frac{DP'_g}{Dt}\right]}$$

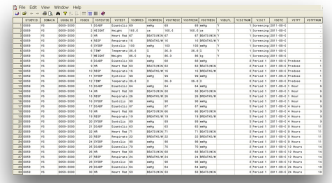
**in reality…**

Problem Dimension $N \to \infty$

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label)



Dataset

$$x \longrightarrow \boxed{f(\cdot)} \longrightarrow y = f(x) + \varepsilon$$

e.g. $f(x) =$

$$\frac{D}{Dt}\overline{w'^i w'^j} + \overline{w'^i w'^\alpha}\nabla_\alpha \bar{u}^j + \overline{w'^j w'^\alpha}\nabla_\alpha \bar{u}^i - \alpha\left(g^{i\alpha}\overline{w'^j\frac{T'}{\bar{T}}} + g^{j\alpha}\overline{w'^i\frac{T'}{\bar{T}}}\right)\left(\nabla_\alpha\bar{\Phi} + \frac{D\bar{u}_\alpha}{Dt}\right)$$

$$+\frac{1}{\bar{\rho}}\nabla_\alpha[\bar{\rho}\overline{u'^\alpha w'^i w'^j} + \overline{(g^{i\alpha}w'^j + g^{j\alpha}w'^i)P'} - \overline{w'^i\sigma^{j\alpha}(u')} - \overline{w'^j\sigma^{i\alpha}(u')}]$$

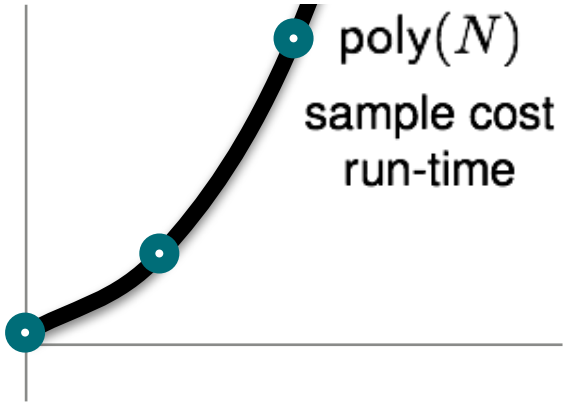
$$+\frac{1}{\bar{\rho}}\overline{w'^i w'^j\nabla_\alpha(\bar{\rho}u'^\alpha)} - \overline{P'(g^{i\alpha}\nabla_\alpha w'^j + g^{j\alpha}\nabla_\alpha w'^i)} = -\frac{1}{\bar{\rho}}[\overline{\sigma^{i\alpha}(u')\nabla_\alpha w'^j} + \overline{\sigma^{j\alpha}(u')\nabla_\alpha w'^i}] = -\epsilon_2^{ij}\,,$$

$$(1+e_4)\frac{D}{Dt}\overline{\left(\frac{T'}{\bar{T}}\right)^2} - 2f(t)\overline{\left(\frac{T'}{\bar{T}}\right)^2} - 2\overline{w'^\alpha\frac{T'}{\bar{T}}}D_\alpha + \frac{1}{(1+e_4)\bar{\rho}C_p^2}\nabla_\alpha\left[(1+e_4)^2C_p^2\bar{\rho}\overline{w'^\alpha\left(\frac{T'}{\bar{T}}\right)^2}\right] + \frac{1+e_4}{\bar{\rho}}\overline{\left(\frac{T'}{\bar{T}}\right)^2\nabla_\alpha(\rho u'^\alpha)}$$

$$+\frac{2}{\bar{\rho}\bar{T}C_p}\overline{\frac{T'}{\bar{T}}\left[P'\nabla_\alpha w'^\alpha - \nabla_\alpha(P'_g w'^\alpha) - \frac{DP'_g}{Dt}\right]} = \frac{2}{\bar{\rho}\bar{T}C_p}\overline{\frac{T'}{\bar{T}}[\sigma^{\alpha\beta}(u')\nabla_\alpha u'_\beta - \nabla_\alpha F'^\alpha_r]} = -\epsilon_2\,,$$

$$(1+e_4)\left[\frac{D}{Dt}\overline{\left(w'^i\frac{T'}{\bar{T}}\right)} + \overline{w'^\alpha\frac{T'}{\bar{T}}}\nabla_\alpha\bar{u}^i - \alpha\overline{\left(\frac{T'}{\bar{T}}\right)^2}g^{i\alpha}\left(\nabla_\alpha\bar{\Phi} + \frac{D\bar{u}_\alpha}{Dt}\right)\right] - f(t)\overline{w'^i\frac{T'}{\bar{T}}} - \overline{w'^i w'^\alpha}D_\alpha$$

$$+\frac{1}{\bar{\rho}C_p}\nabla_\alpha\left[(1+e_4)C_p\bar{\rho}\overline{w'^i w'^\alpha\frac{T'}{\bar{T}}}\right] + \frac{1+e_4}{\bar{\rho}}\overline{w'^i\frac{T'}{\bar{T}}\nabla_\alpha(\rho u'^\alpha)} + \frac{1}{\bar{\rho}\bar{T}C_p}\overline{w'^i\left[P'\nabla_\alpha w'^\alpha - \nabla_\alpha(P'_g w'^\alpha) - \frac{DP'_g}{Dt}\right]}$$



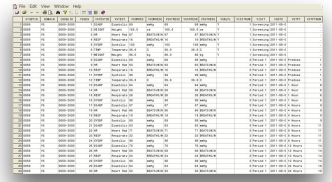

Problem Dimension $N$

Problem Dimension $N \to \infty$

# Motivation

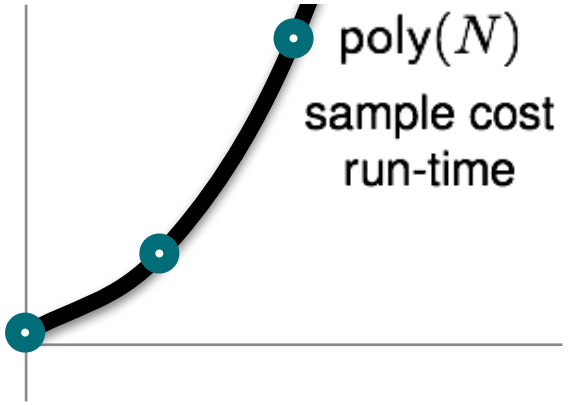
- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

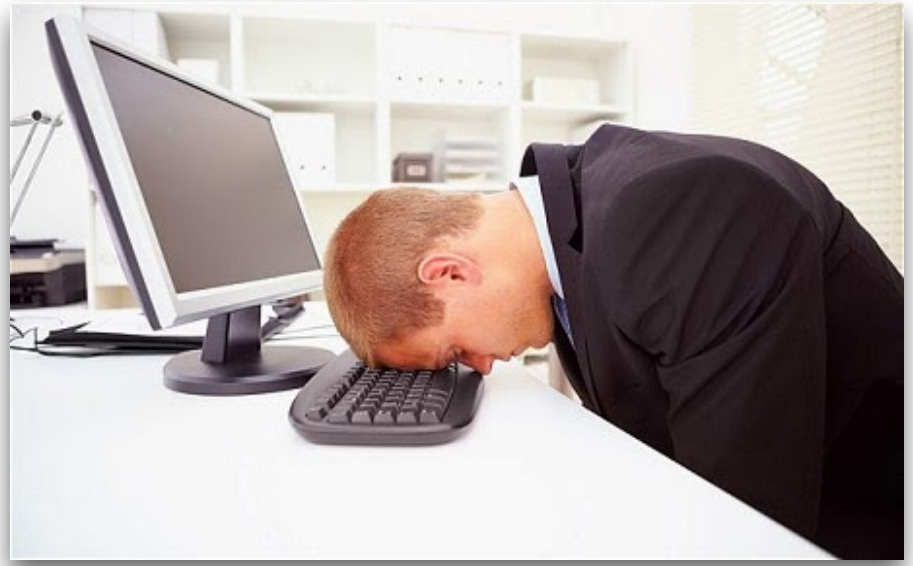
$(x$ : feature, $y$ : label$)$



Dataset

$$x \longrightarrow \boxed{f(\cdot)} \longrightarrow y = f(x) + \varepsilon$$

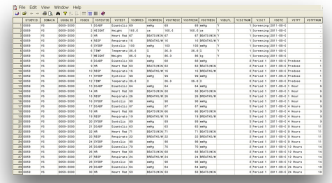


# Motivation

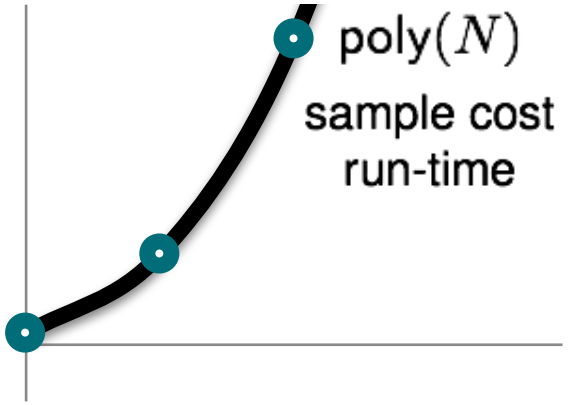
- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

$(x$ : feature, $y$ : label$)$



Dataset

$$x \longrightarrow \boxed{f(\cdot)} \longrightarrow y = f(x) + \varepsilon$$

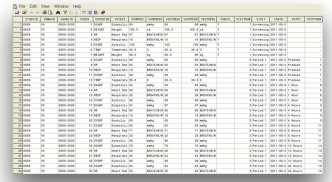



What **if**

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

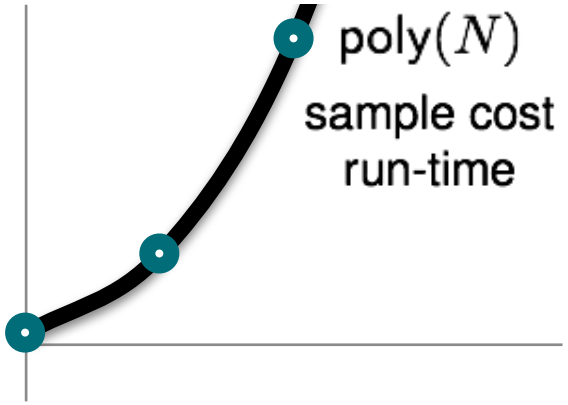
$(x$ : feature, $y$ : label$)$



Dataset

$f(\cdot)$

$$y = f(x) + \varepsilon$$

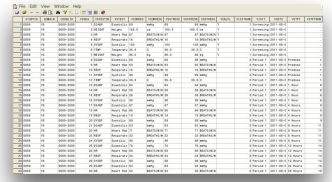



What **if**
— we can **actively choose** training data

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

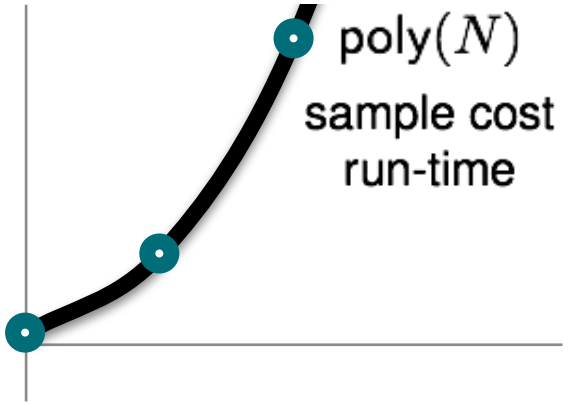
$(x$ : feature, $y$ : label$)$



Dataset

$f(\cdot)$

$$y = f(x) + \varepsilon$$

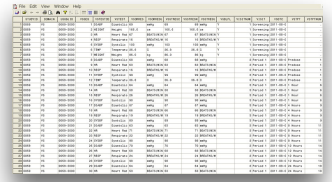



What **if**
— we can **actively choose** training data
— the model has **sublinear d.o.f**

# Motivation

- Given training data points $(x, y)$, our goal is to learn
  - **a certain rule** $f$ that explains the label $y$ based on features $x$:

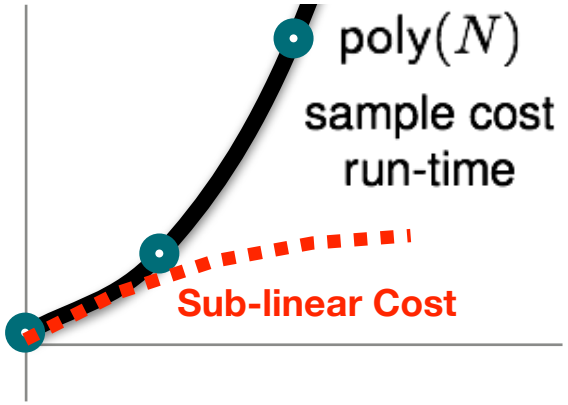
$(x$ : feature, $y$ : label$)$



Dataset

$f(\cdot)$

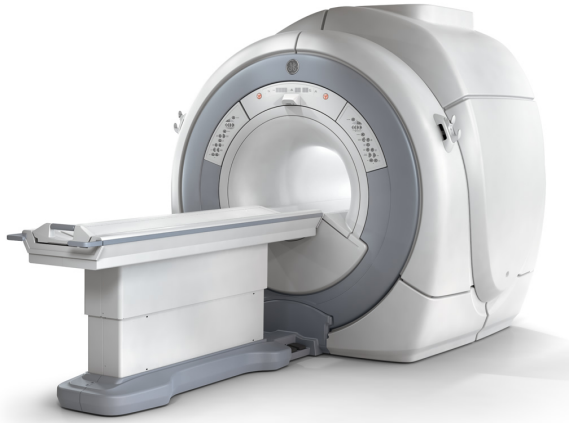
$$y = f(x) + \varepsilon$$




What **if**
— we can **actively choose** training data
— the model has **sublinear d.o.f.**

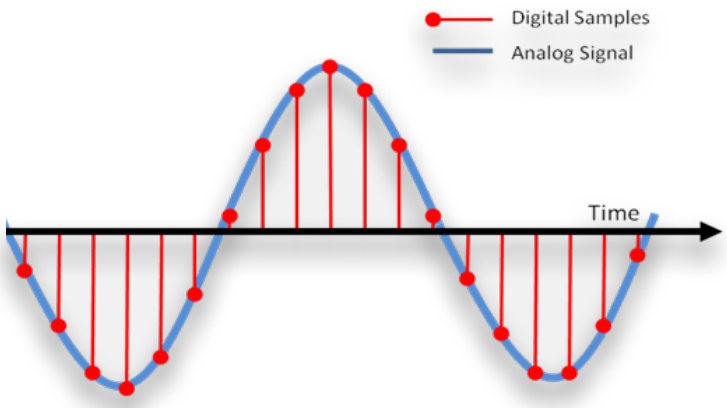
**Can we achieve fast & robust learning with active sampling + coding theory?**

# Potential Impact

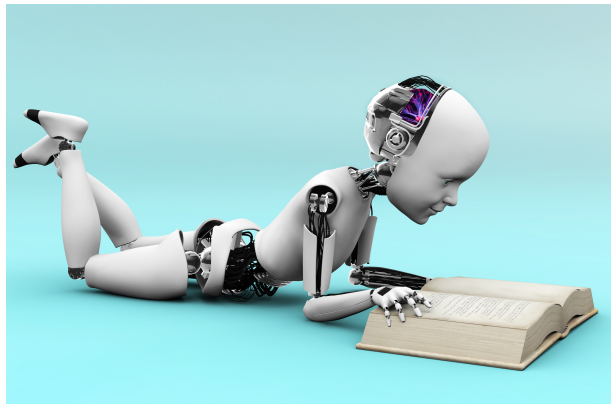


**MRI**

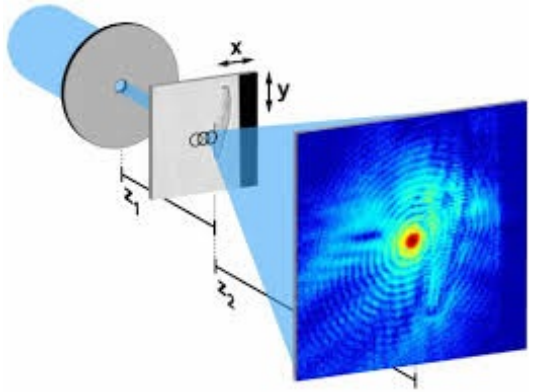
**Radio Astronomy**



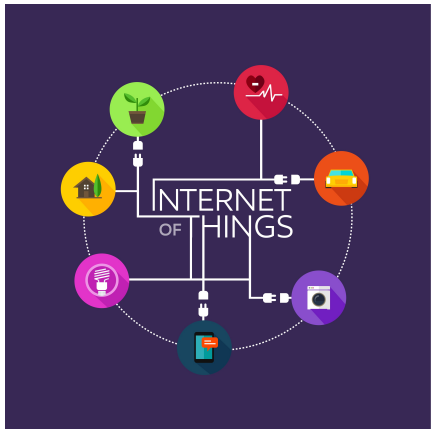

**Sub-Nyquist Sampling**



**Machine Learning**




**Computational Imaging**




**IoT**

Sparse Spectrum (DFT/WHT)

Sparse-graph codes

Sameer Pawar

Xiao (Simon) Li

Orhan Ocal

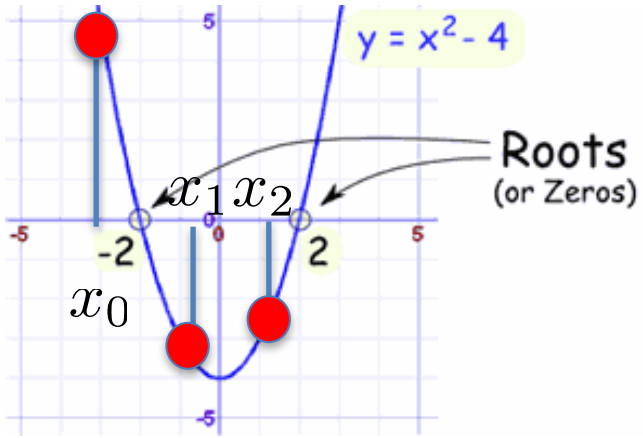
# Polynomial Learning: High-School Algebra Edition

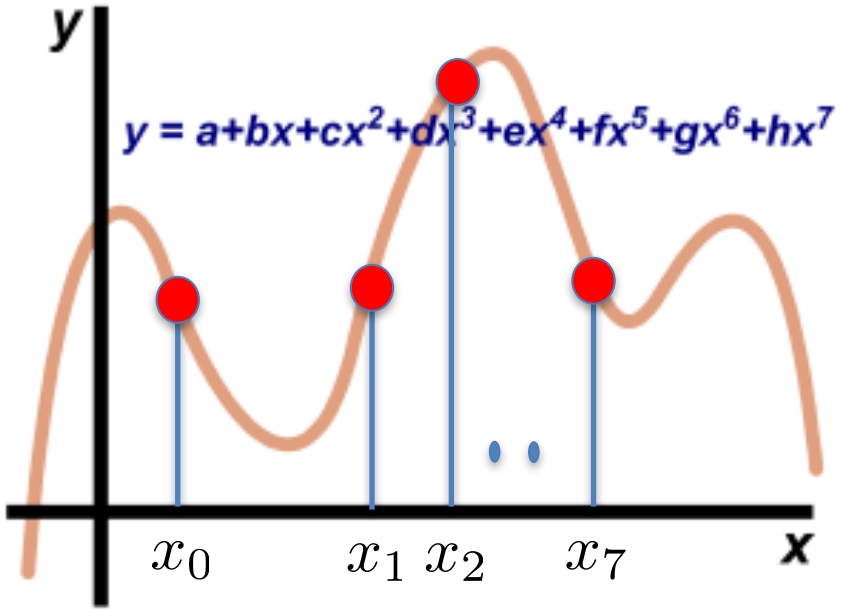
- Given $f(x) = \sum_{n=0}^{N-1} F_n x^n$
- Find coefficients $\{F_n\}_{n=0}^{N-1}$

$$x_i \longrightarrow \boxed{f(x)} \longrightarrow f(x_i)$$

*Q. How many evaluations do we need?*

***A. N evaluations***



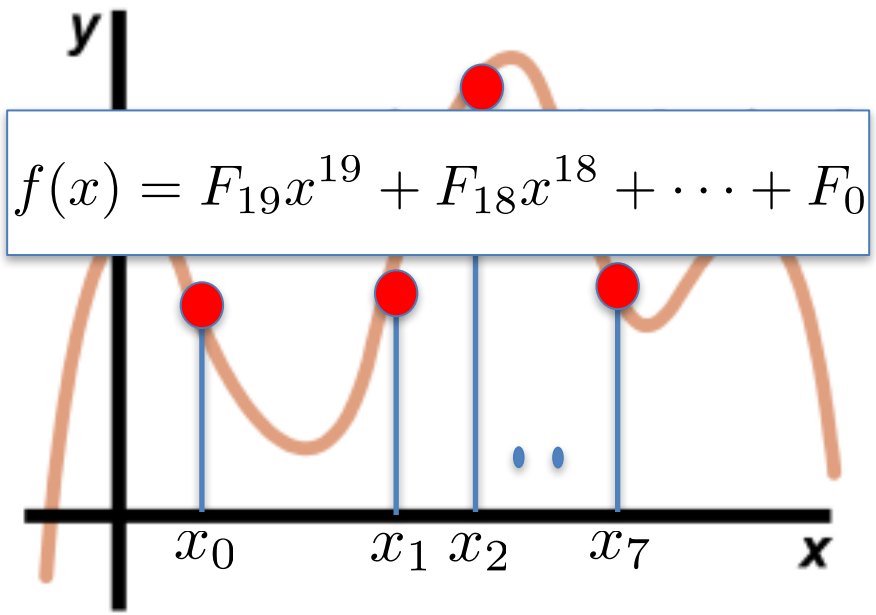



# Recovering the coefficients

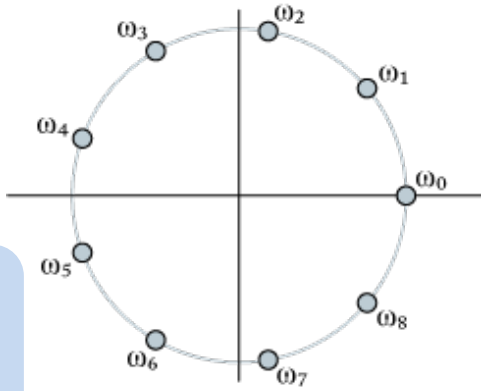
- Given $f(x) = \sum_{n=0}^{N-1} F_n x^n$
- Find coefficients $\{F_n\}_{n=0}^{N-1}$

$$x_i \longrightarrow \boxed{f(x)} \longrightarrow f(x_i)$$

$$f(x) = F_{19}x^{19} + F_{18}x^{18} + \cdots + F_0$$




$$\begin{bmatrix} f(X_0) \\ f(X_1) \\ f(X_2) \\ \vdots \\ f(X_{19}) \end{bmatrix} = \begin{bmatrix} 1 & X_0 & \cdots & X_0^{19} \\ 1 & X_1 & \cdots & X_1^{19} \\ 1 & X_2 & \cdots & X_2^{19} \\ \vdots & & & \\ 1 & X_{19} & \cdots & X_{19}^{19} \end{bmatrix} \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ \vdots \\ F_{19} \end{bmatrix}$$

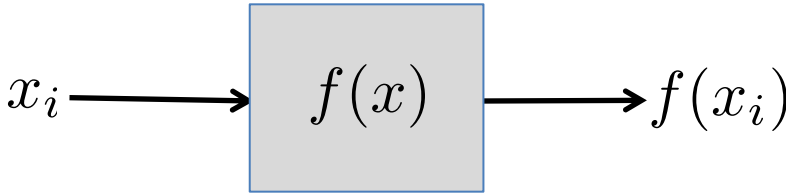


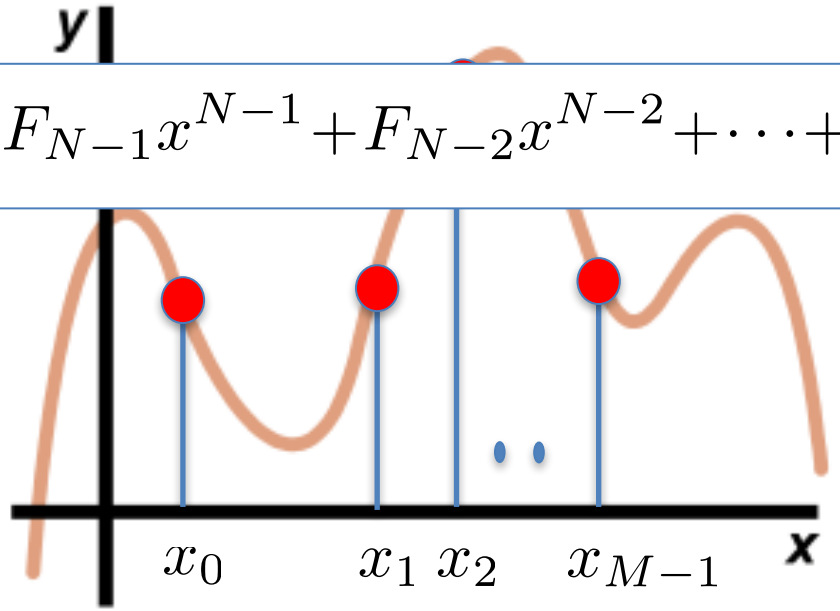

inverse Discrete Fourier Transform (DFT)

if $X_m = e^{\mathrm{i}\frac{2\pi}{N}m}$

# What if only **K** of **N** coeffs. non-zero?

**K** sublinear in **N** (**K**/**N**➔0)




$$f(x) = F_{N-1}x^{N-1} + F_{N-2}x^{N-2} + \cdots + F_0$$

**Example:**

Degree **N= 1 million**

Sparsity **K = 200**

***(spoiler alert)***

*# evalulations = **616***

# Problem Formulation

Compute the $K$-sparse DFT of $x \in \mathbb{C}^N$ with $K \ll N$:

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k \in \mathcal{K}} X[k] e^{i \frac{2\pi k}{N} n} \qquad n = 0, \cdots, N-1$$

Support $\mathcal{K}$ chosen from $[N]$ uniformly at random

# Results: FFAST

## (Fast Fourier Aliasing-Based Sparse Transform)

- Noiseless: For **K** sublinear in **N**
  - Can compute **K**-sparse **N**-length DFT
  - Using **fewer than 4K** samples in
  - **O(K log K)** time

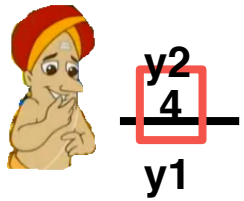
Robust to noise: same framework: **O(K log$^{4/3}$ N)** samples in **O(K log$^{7/3}$ N)** time

Sub-linear time recovery when d.o.f. sublinear!

Pawar, **R**, IEEE Trans. Inf. Theory , 2018

# Puzzle: Gold thief


- One unknown thief.
- Steals unknown but fixed amount from each coin.
- What is min. no. of weighings needed ?
  - 2 are enough.

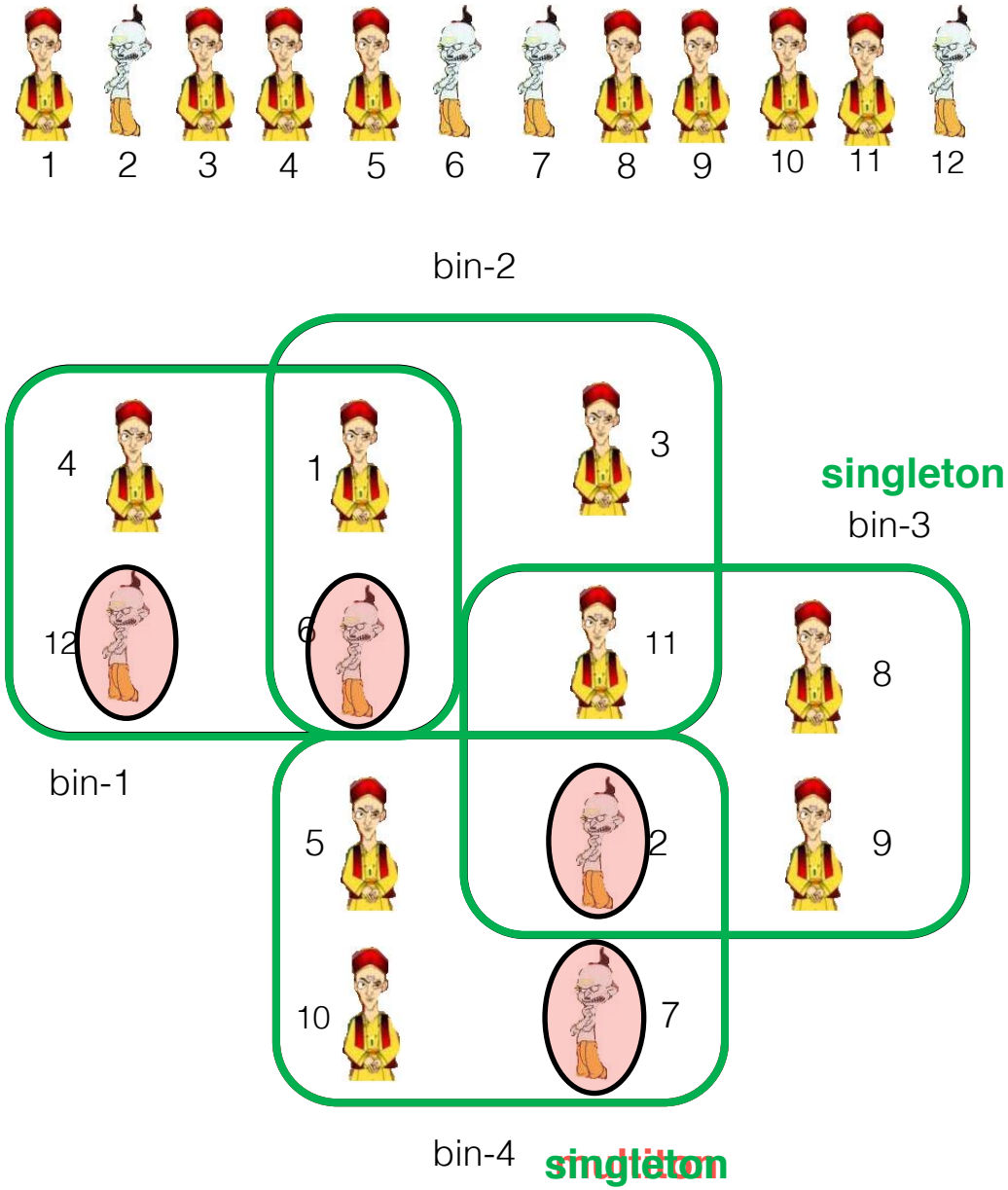



$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} = \begin{bmatrix} -5 \\ -20 \end{bmatrix} = \begin{bmatrix} y1 \\ y2 \end{bmatrix}$$

***Differential weight***

**Ratio-test identifies the location**

# 4-thieves among 12-treasurers




**Key Ideas:**

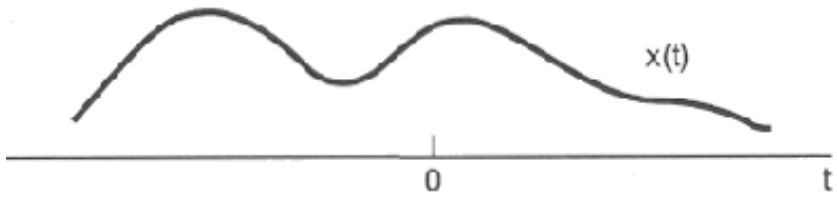
1. Randomly group the treasurers.
2. If there is a single thief problem
   - ✓ Ratio test
   - ✓ Iterate

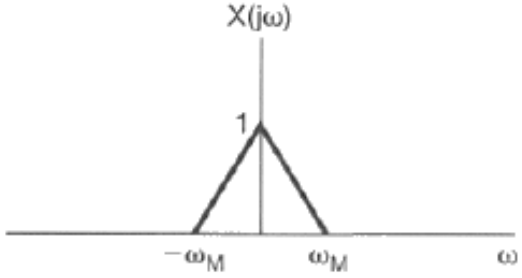
**Questions:**

1. How many groups needed?
2. How to form groups?
3. How to identify if a group has a single thief?

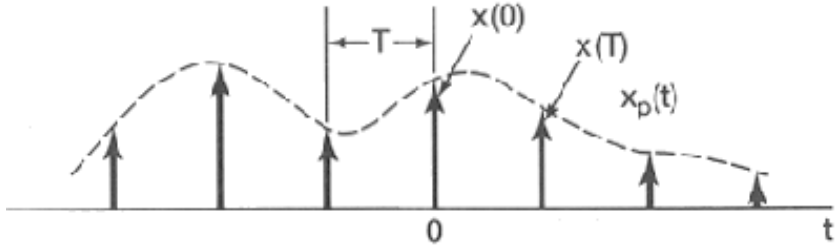
# Aliasing

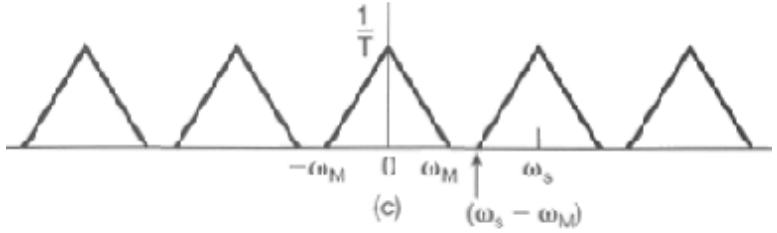
## Signal and its spectrum



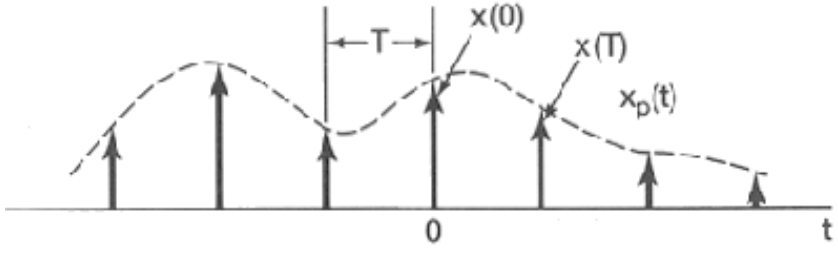



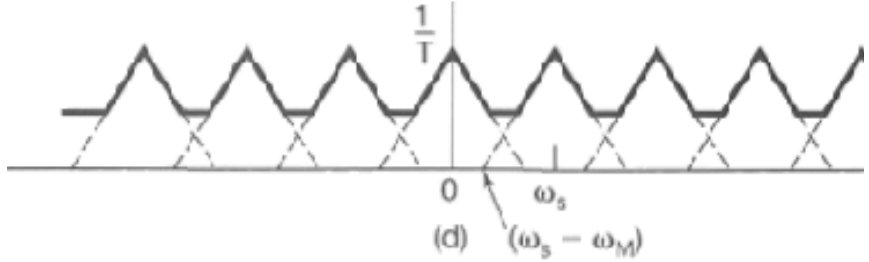

## Sampling




## Sub-sampling



# Insights

**Sub-sampling**
below Nyquist rate

→ **Aliasing** in the frequency domain

Clever sub-sampling
(for **sparse** case)

→ **Good** "alias" code

***Chinese-Remainder-Theorem guided subsampling***

***Sparse graph codes***

## We use coding-theoretic tools

**Design:**

- Randomized constructions of good sparse-graph codes

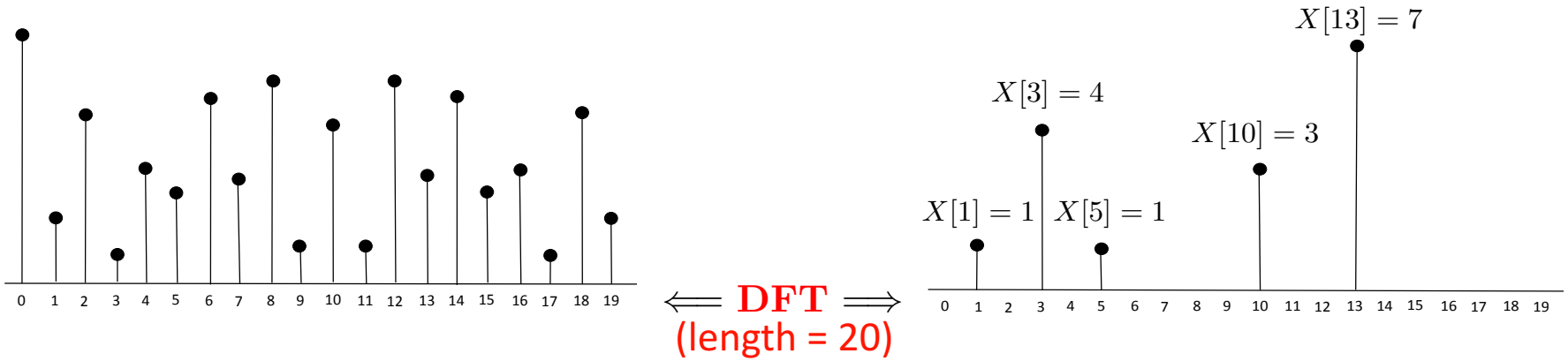
**Analysis:**

- Density evolution
- Martingale
- Expander graph theory

# Main idea

time-domain $x[n]$ length $N = 20$

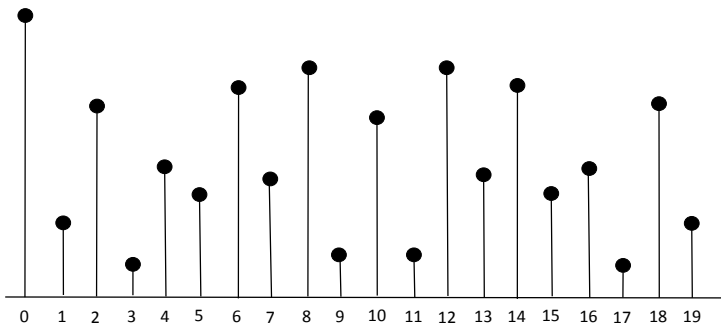
frequency-domain $X[k]$ sparsity $K = 5$



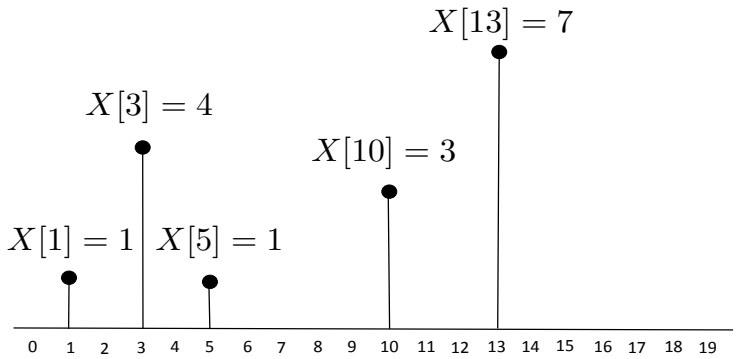
# Main idea

time-domain $x[n]$ length $N = 20$

frequency-domain $X[k]$ sparsity $K = 5$

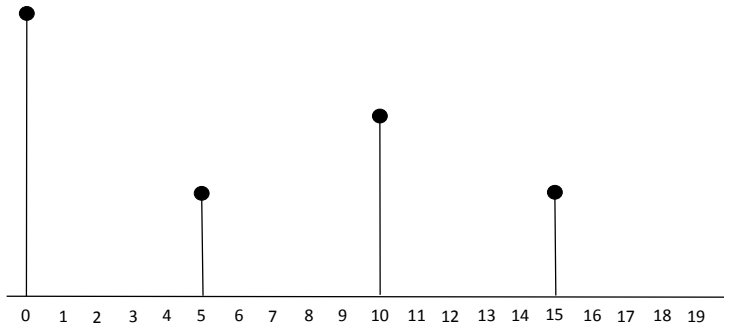



$\Longleftarrow$ **DFT** $\Longrightarrow$
(length = 20)




$\downarrow 5$

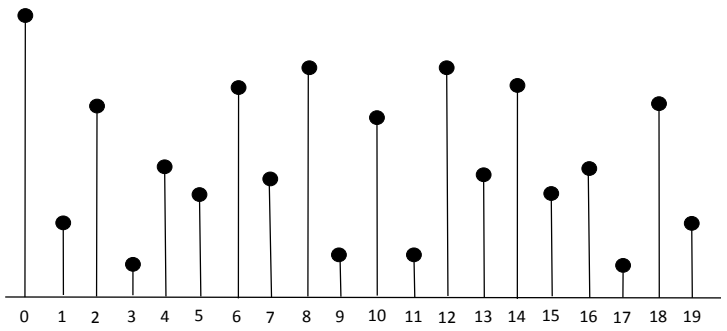
**subsample by 5**



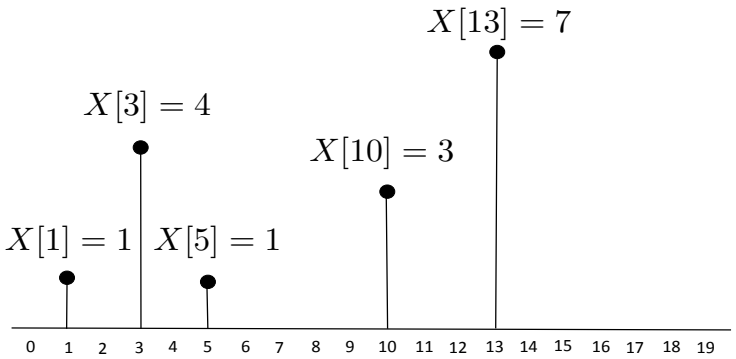
# Main idea

time-domain $x[n]$ length $N = 20$

frequency-domain $X[k]$ sparsity $K = 5$

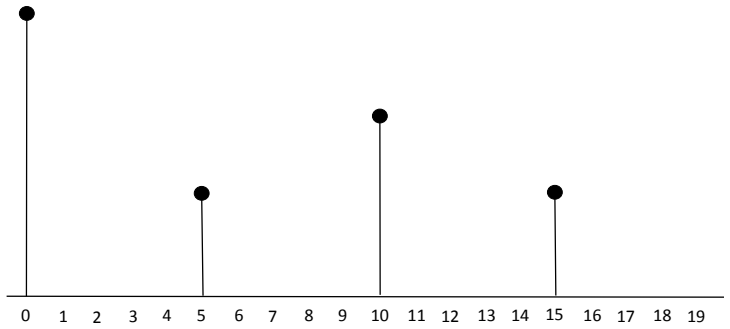



$\Longleftarrow$ **DFT** $\Longrightarrow$
(length = 20)




$\downarrow 5$

**subsample by 5**




$\Longleftarrow$ **DFT** $\Longrightarrow$
(length = 4)

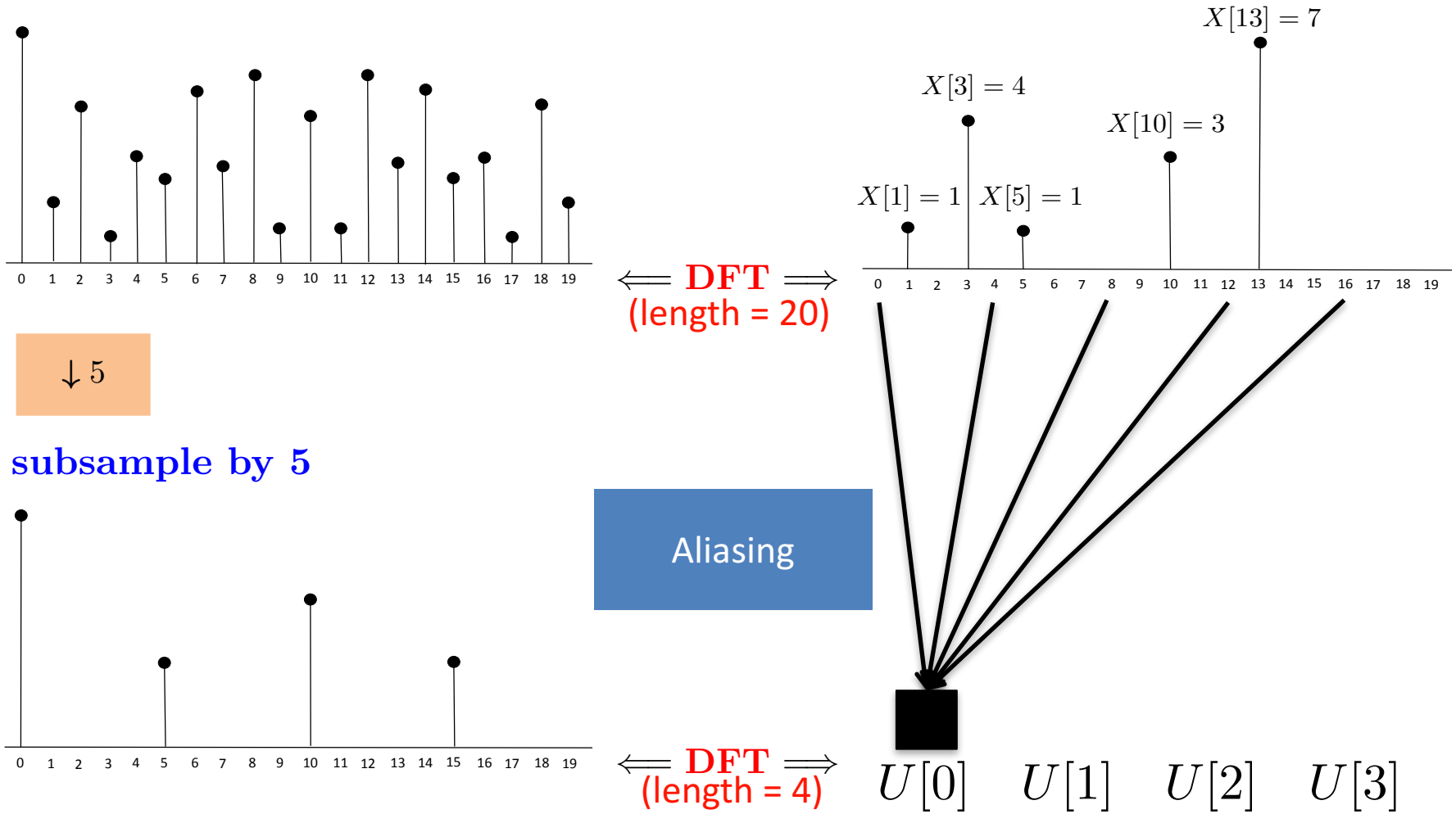
**Our Measurements**

$U[0] \quad U[1] \quad U[2] \quad U[3]$

# Main idea

time-domain $x[n]$ length $N = 20$

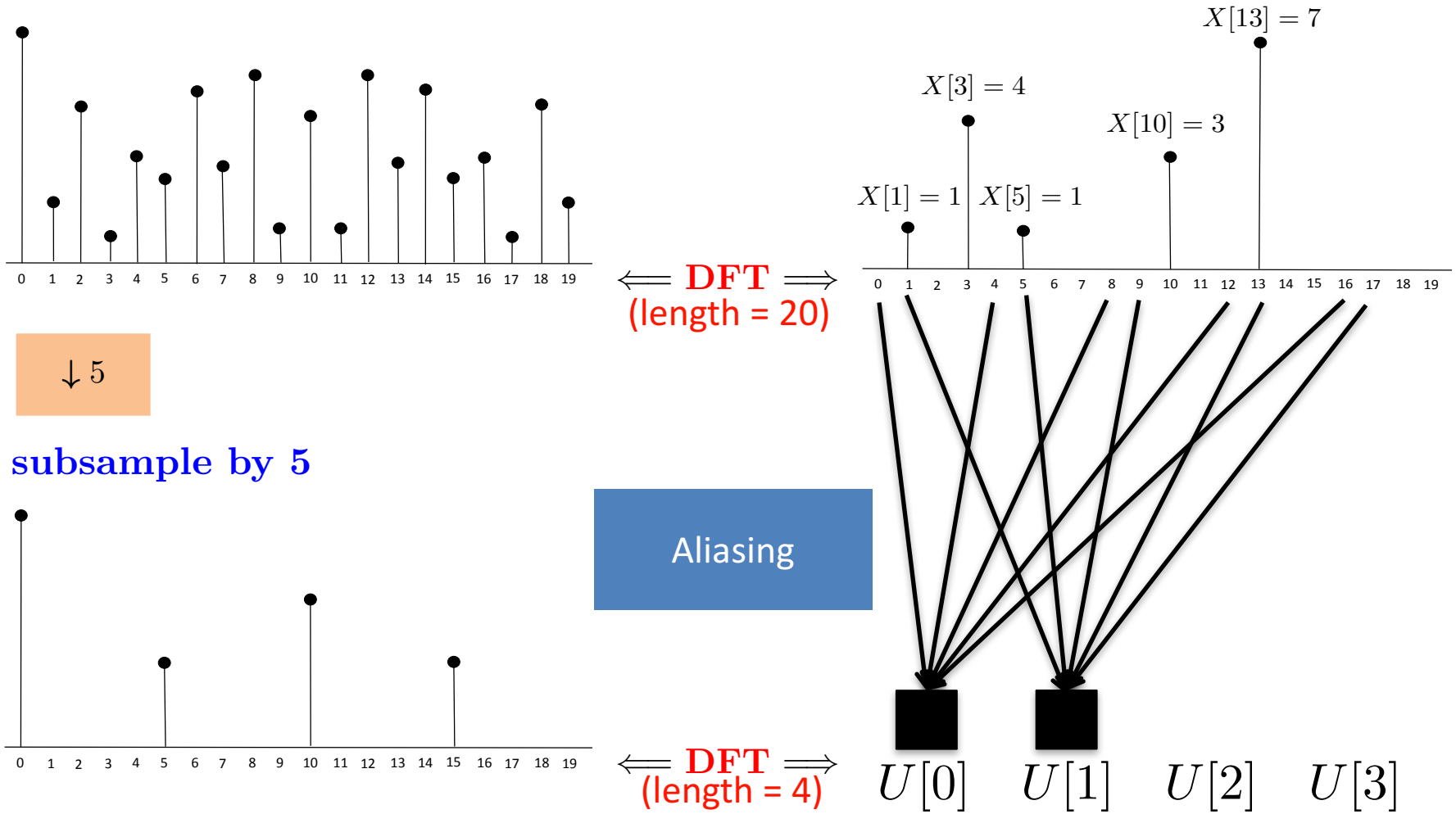
frequency-domain $X[k]$ sparsity $K = 5$



# Main idea

time-domain $x[n]$ length $N = 20$

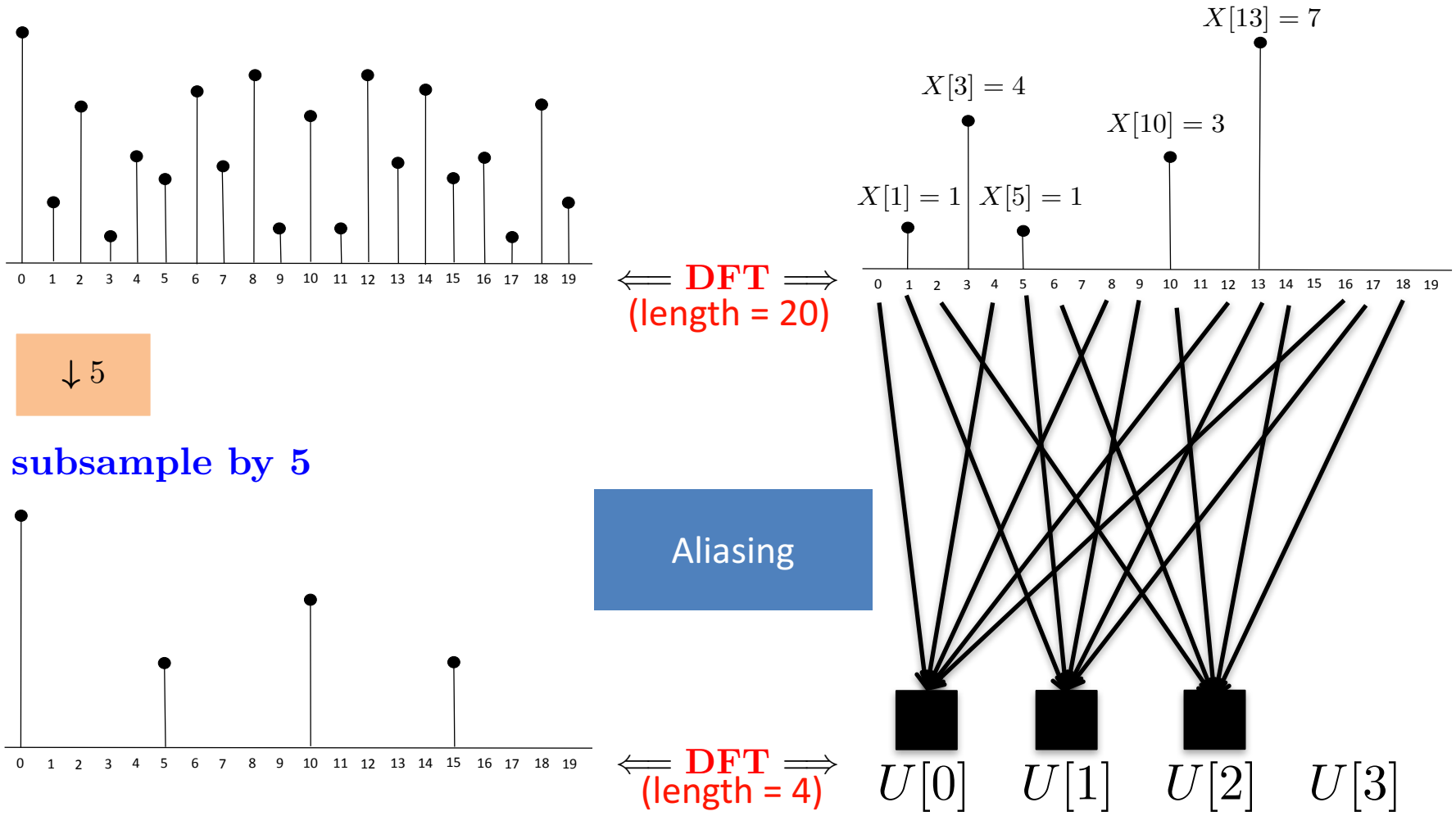
frequency-domain $X[k]$ sparsity $K = 5$



# Main idea

time-domain $x[n]$ length $N = 20$

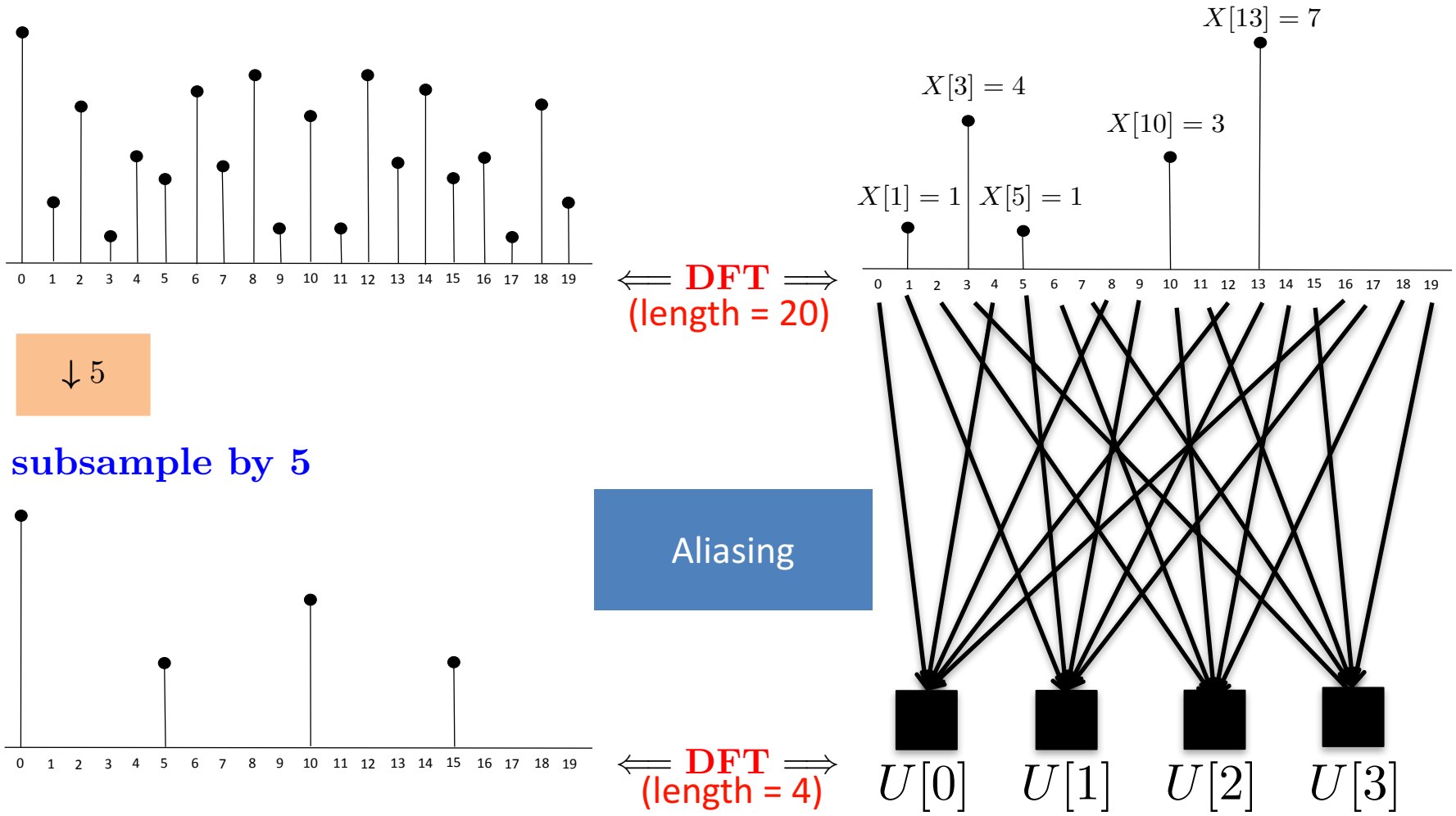
frequency-domain $X[k]$ sparsity $K = 5$



# Main idea

time-domain $x[n]$ length $N = 20$

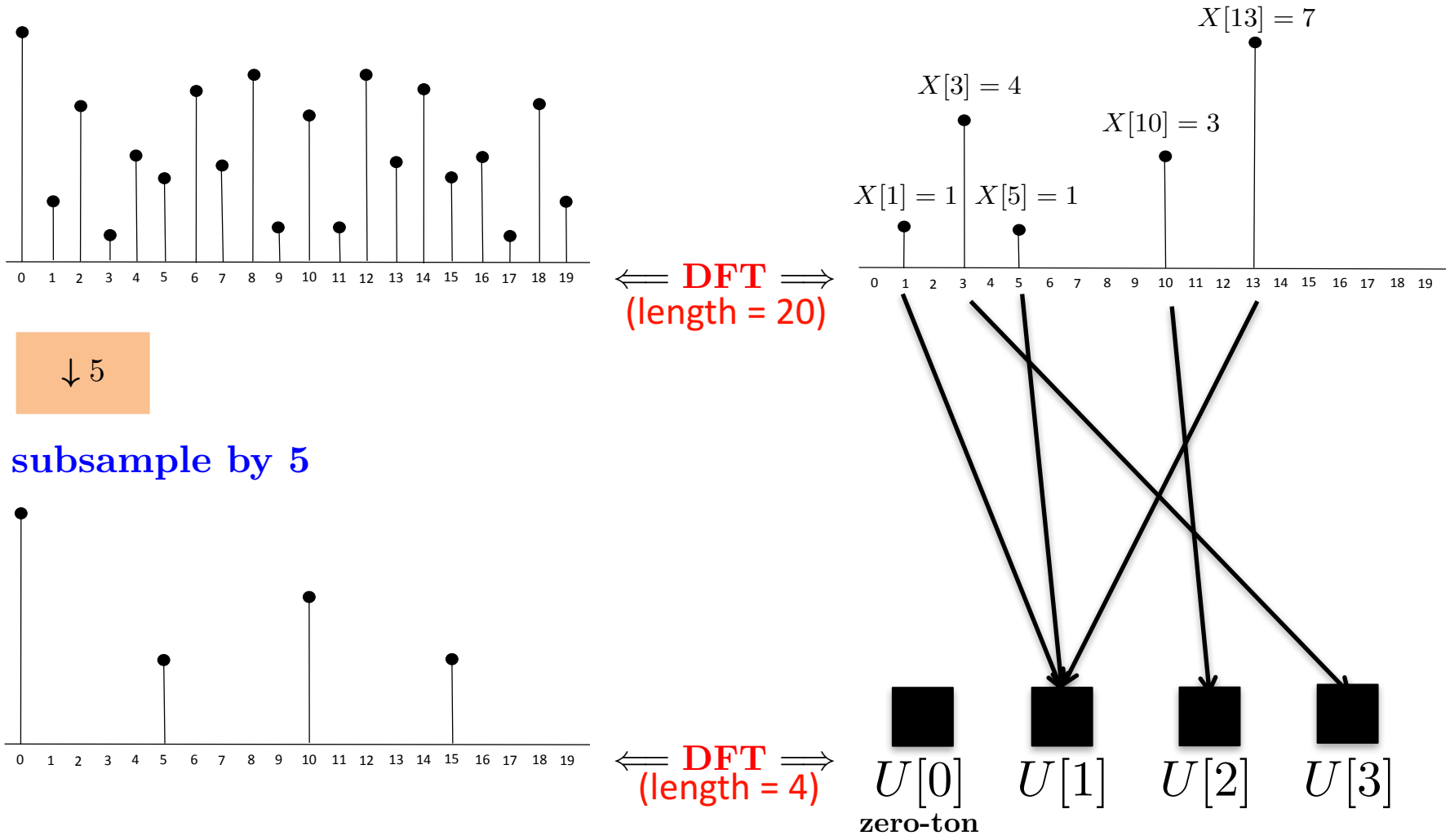
frequency-domain $X[k]$ sparsity $K = 5$



# Main idea

time-domain $x[n]$ length $N = 20$

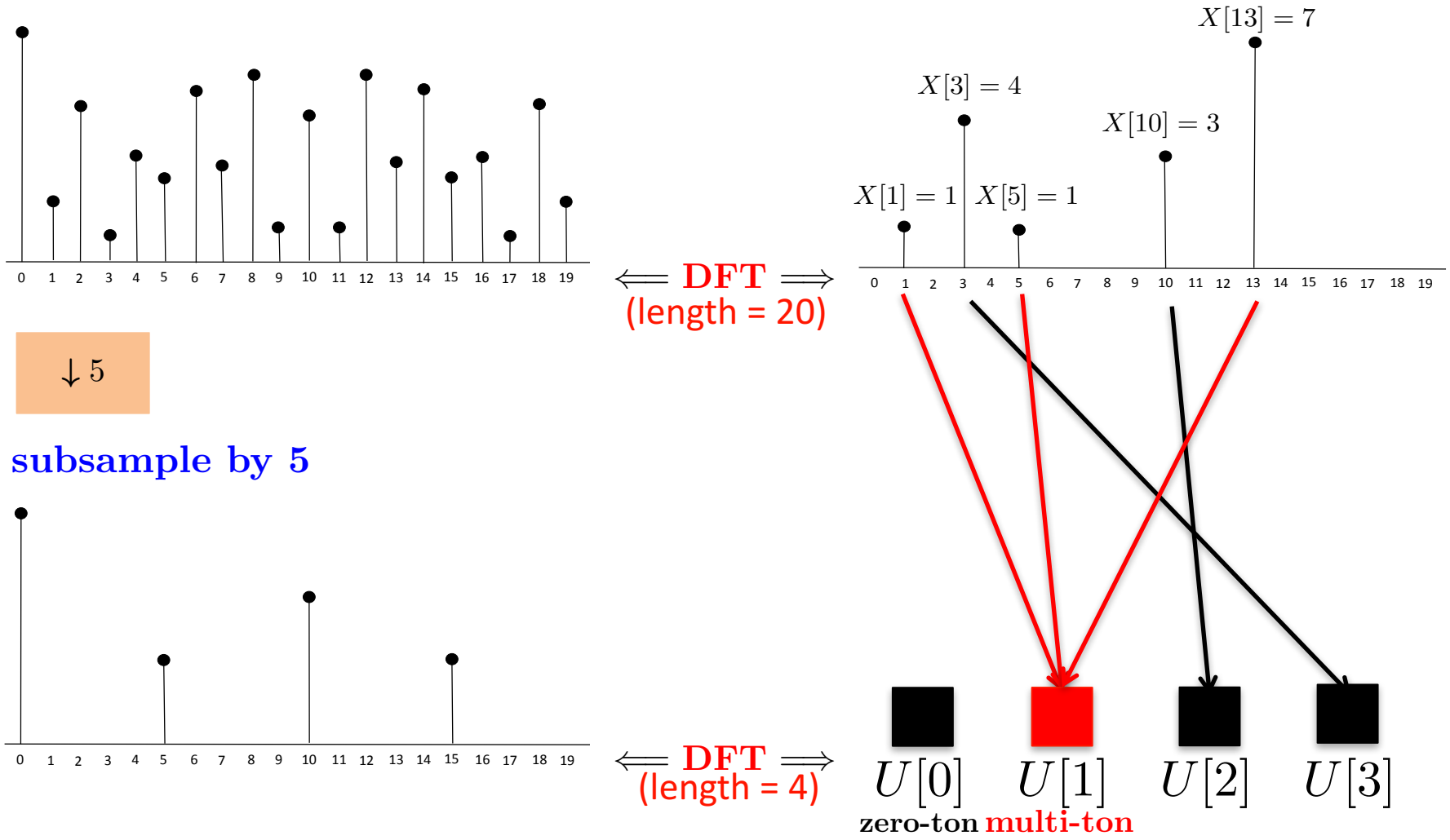
frequency-domain $X[k]$ sparsity $K = 5$



# Main idea

time-domain $x[n]$ length $N = 20$

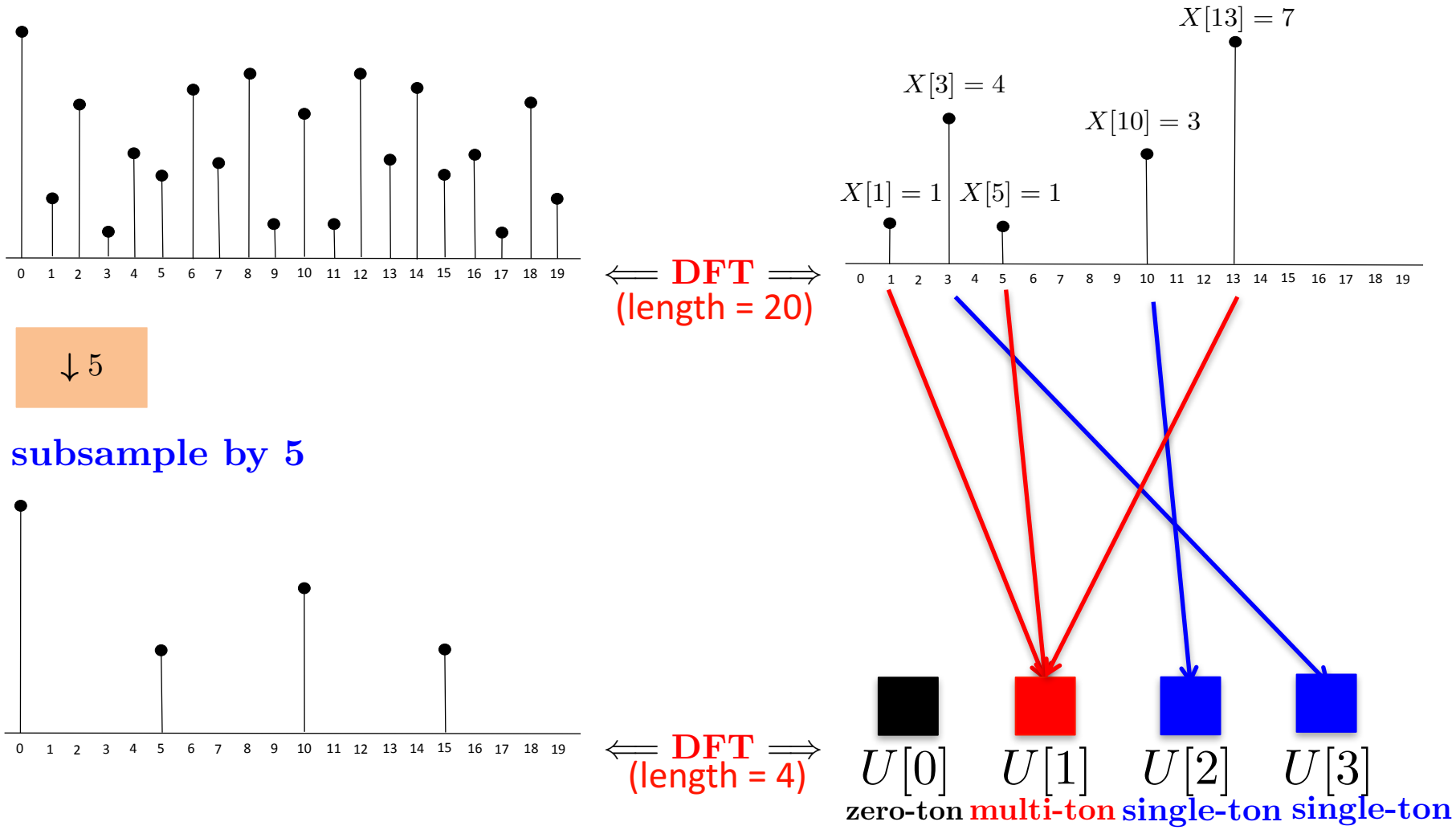
frequency-domain $X[k]$ sparsity $K = 5$



# Main idea

time-domain $x[n]$ length $N = 20$

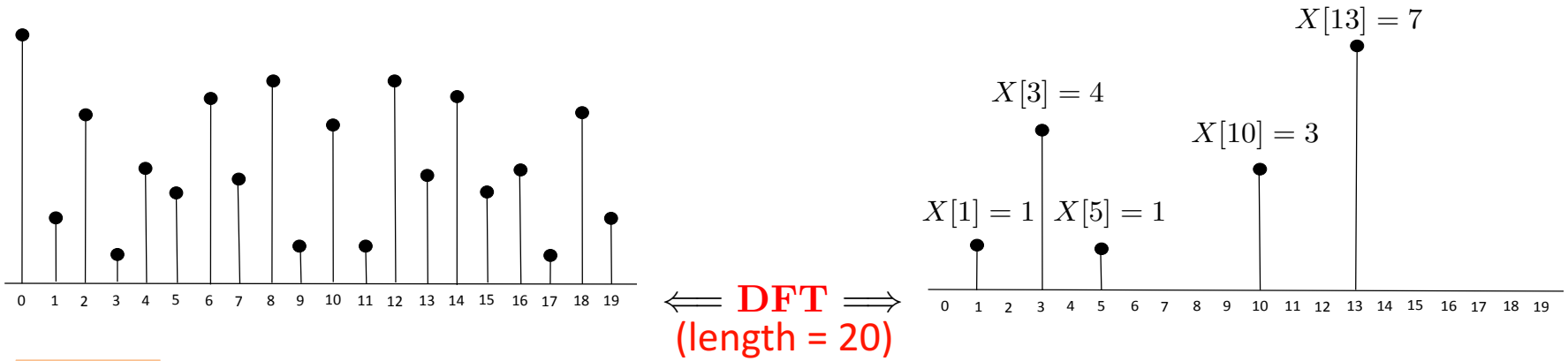
frequency-domain $X[k]$ sparsity $K = 5$



# Main idea

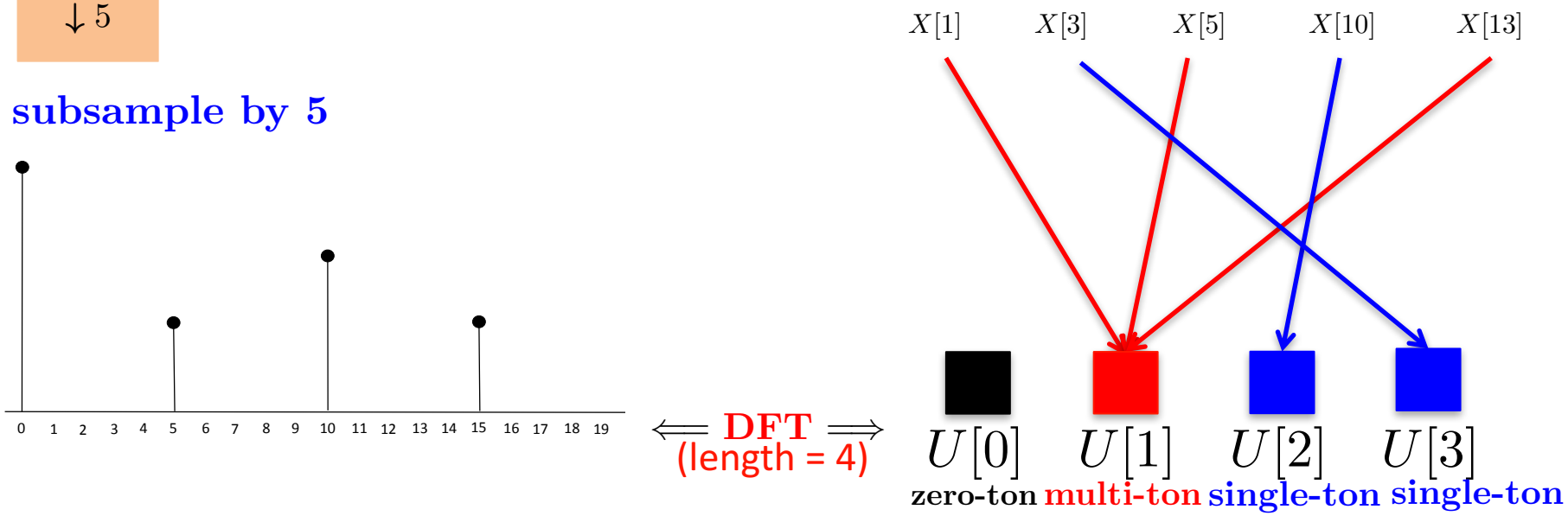
time-domain $x[n]$ length $N = 20$

frequency-domain $X[k]$ sparsity $K = 5$




$\downarrow 5$

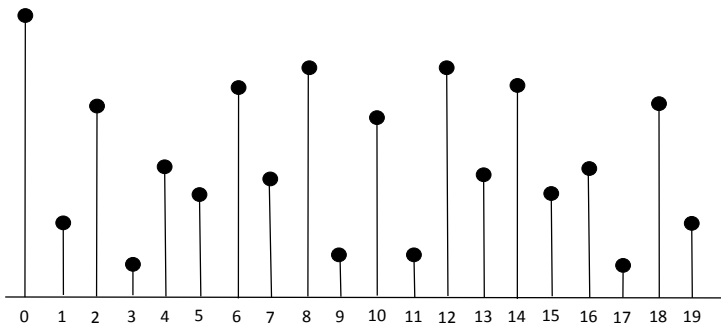
**subsample by 5**



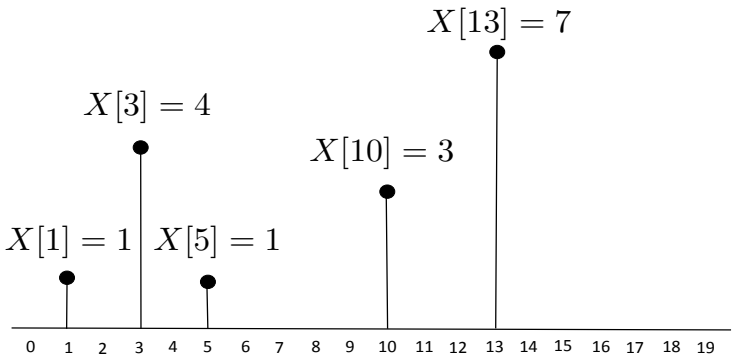
# Main idea

time-domain $x[n]$ length $N = 20$

frequency-domain $X[k]$ sparsity $K = 5$



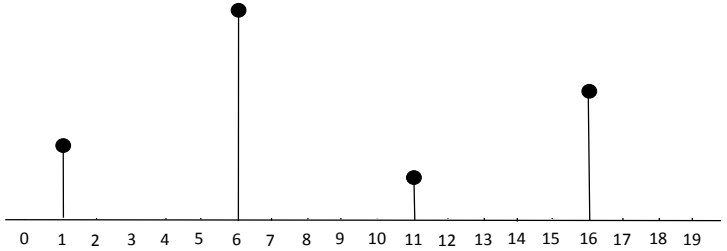

$\Longleftarrow$ **DFT** $\Longrightarrow$
(length = 20)




$X[1]$ $X[3]$ $X[5]$ $X[10]$ $X[13]$

$\downarrow 5$

**shift & subsample by 5**




$\Longleftarrow$ **DFT** $\Longrightarrow$
(length = 4)

**Our Measurements**

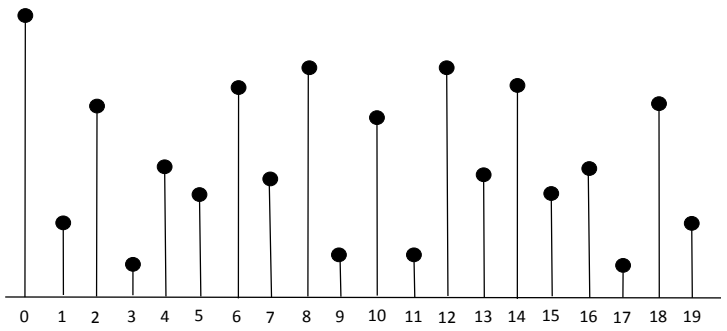
$U_S[0]$ $U_S[1]$ $U_S[2]$ $U_S[3]$

subscript $U_S$ suggests **shift**

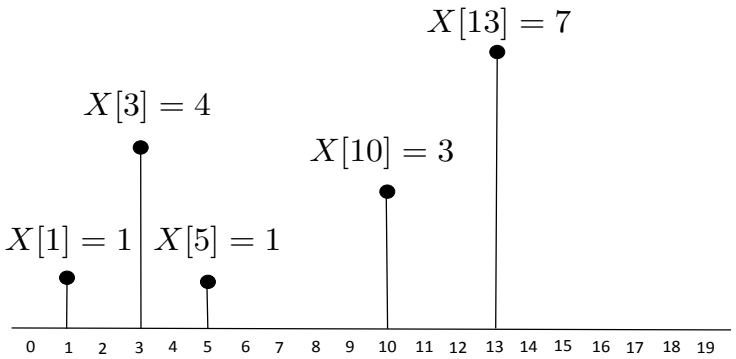
# Main idea

time-domain $x[n]$ length $N = 20$

frequency-domain $X[k]$ sparsity $K = 5$



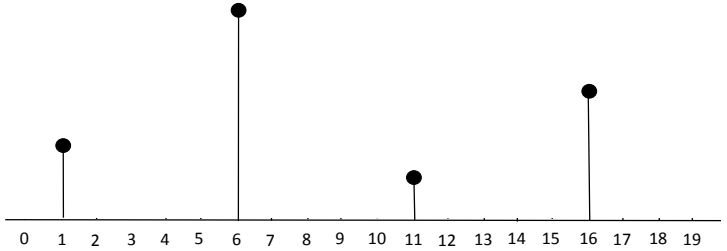

$\Longleftarrow$ **DFT** $\Longrightarrow$
(length = 20)



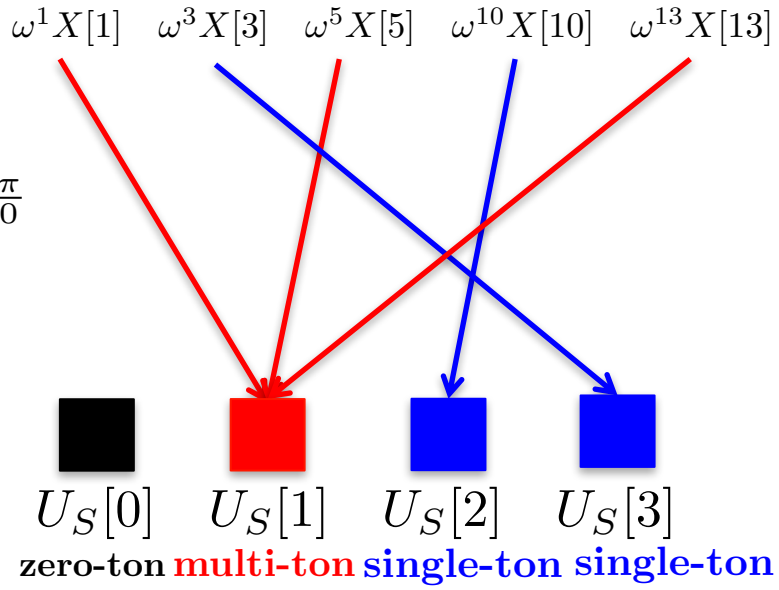

$\downarrow 5$

**shift & subsample by 5**

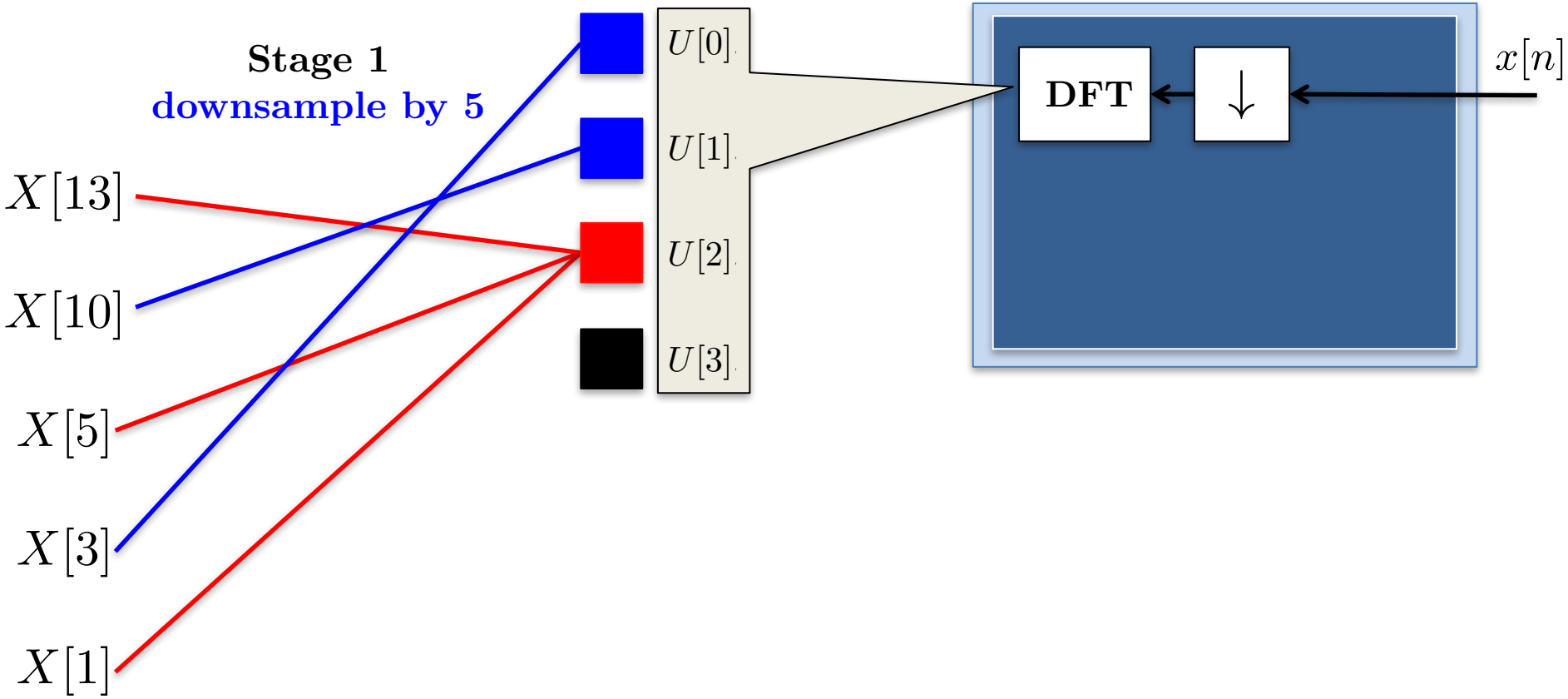
$$\omega = e^{-j\frac{2\pi}{20}}$$



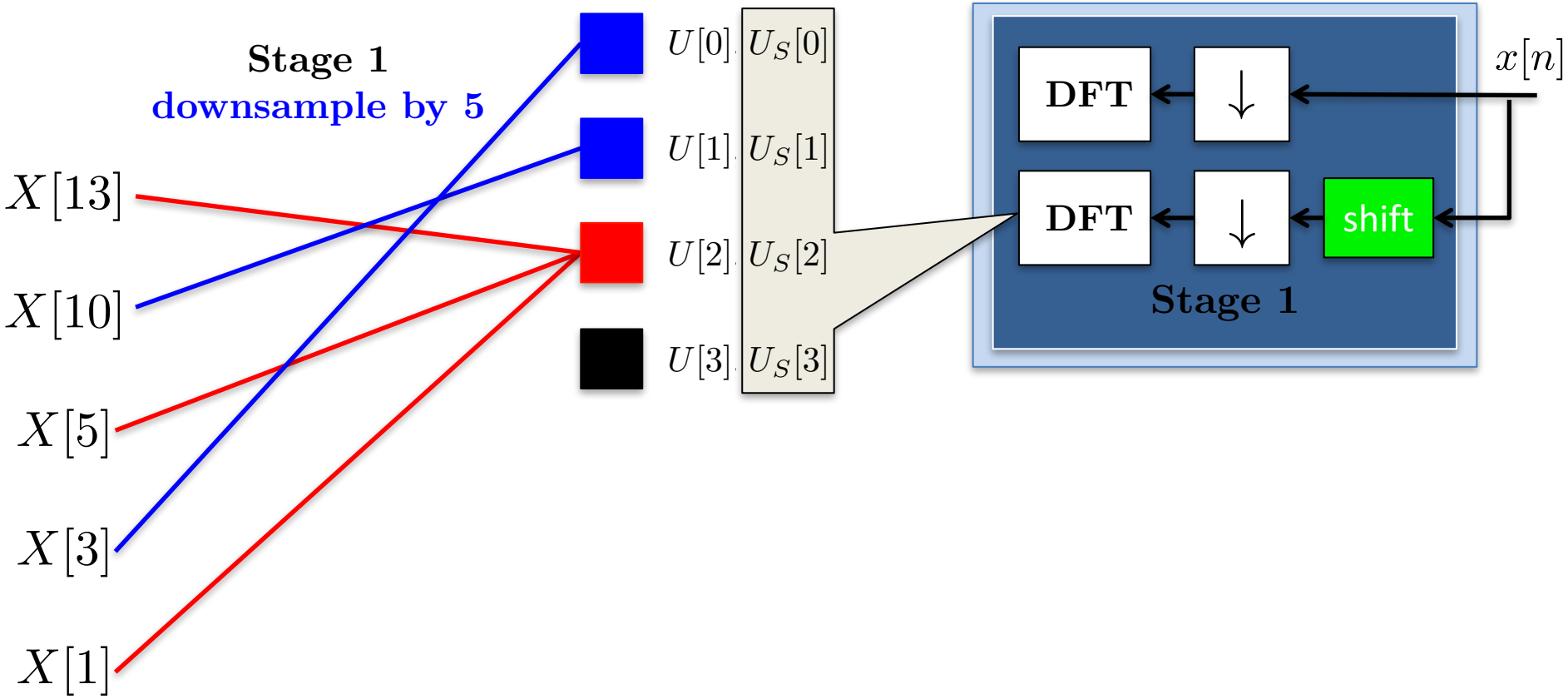

$\Longleftarrow$ **DFT** $\Longrightarrow$
(length = 4)



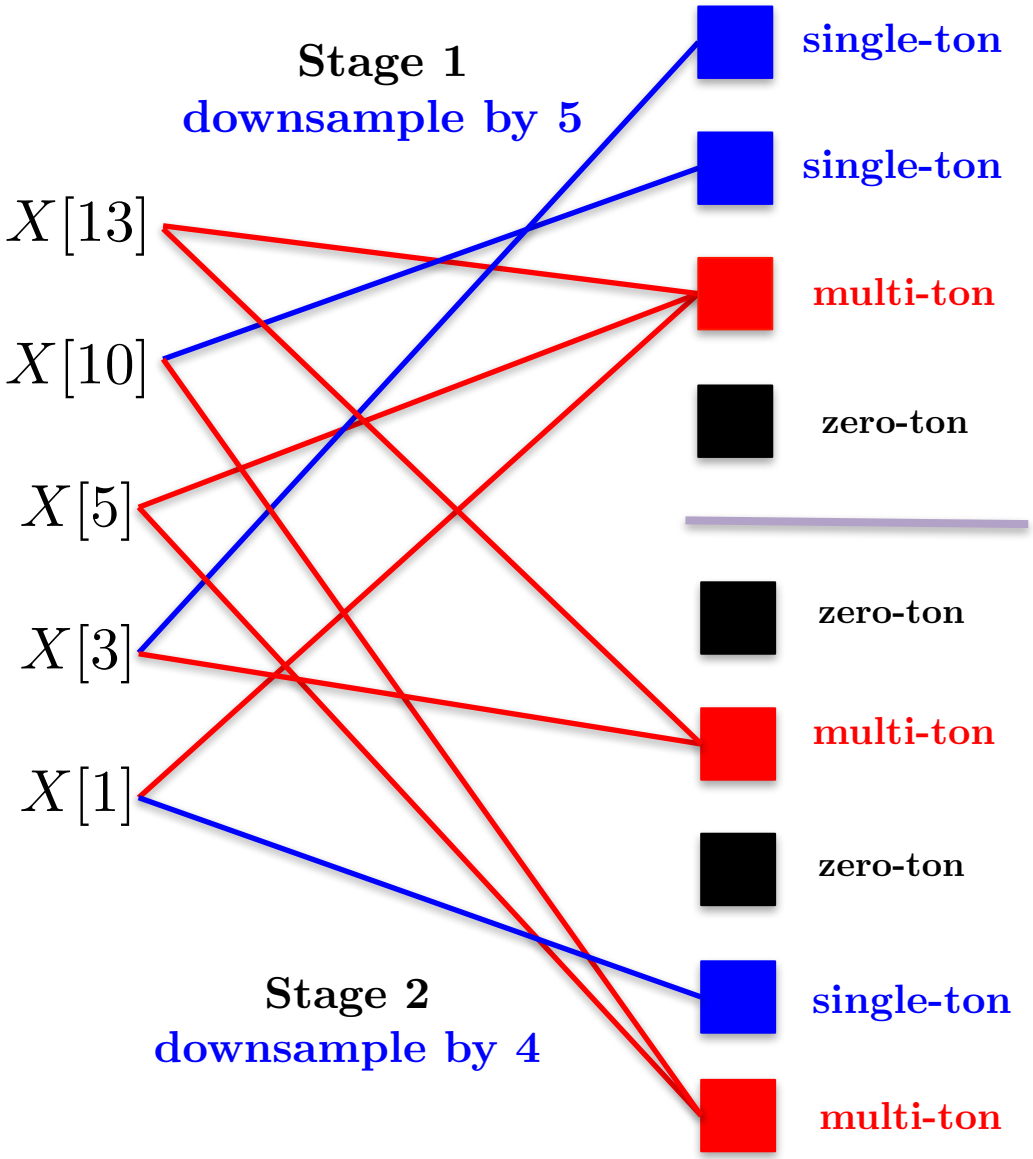
# Main idea

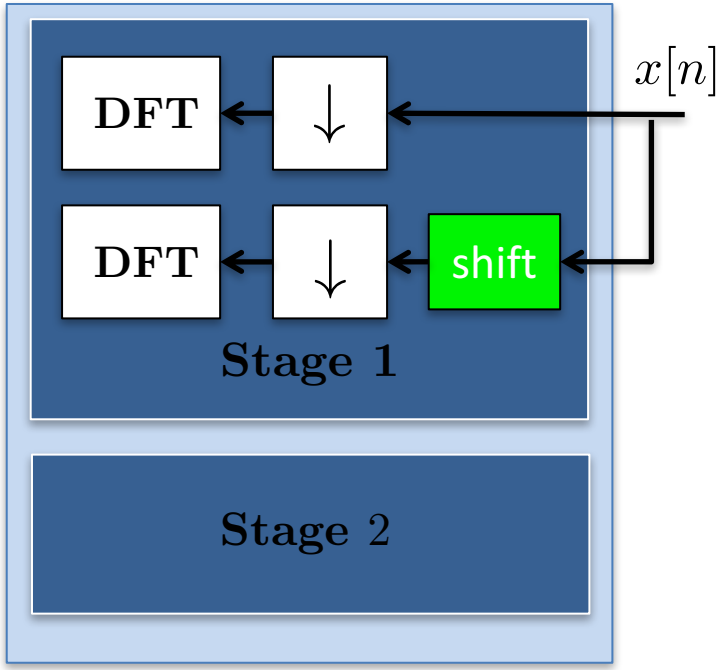

Stage 1
downsample by 5
$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$
$U[0]$
$U[1]$
$U[2]$
$U[3]$
DFT
$\downarrow$
$x[n]$

# Main idea

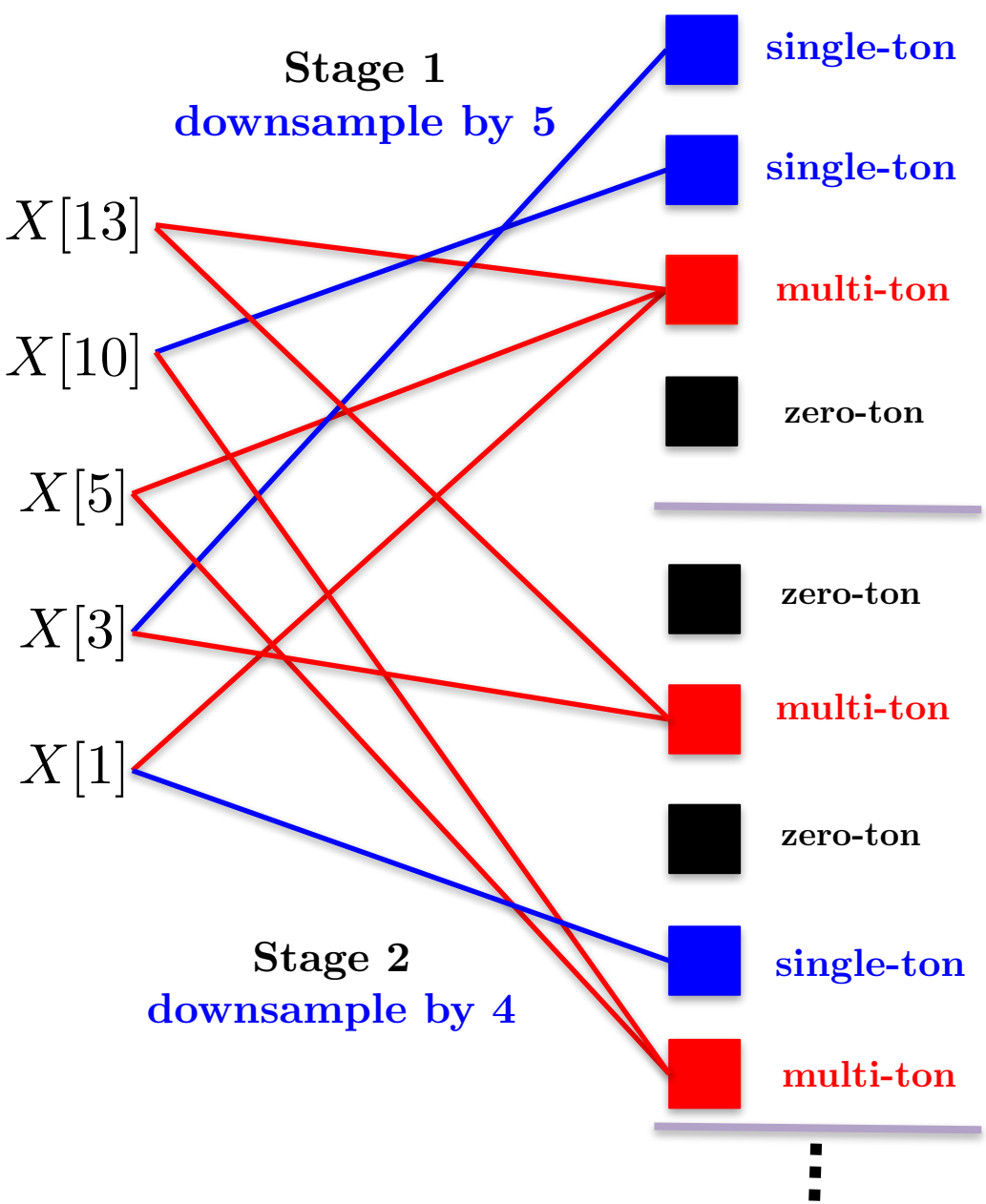

Stage 1
downsample by 5
$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$
$U[0]$
$U[1]$
$U[2]$
$U[3]$
$U_S[0]$
$U_S[1]$
$U_S[2]$
$U_S[3]$
DFT
DFT
shift
Stage 1
$x[n]$

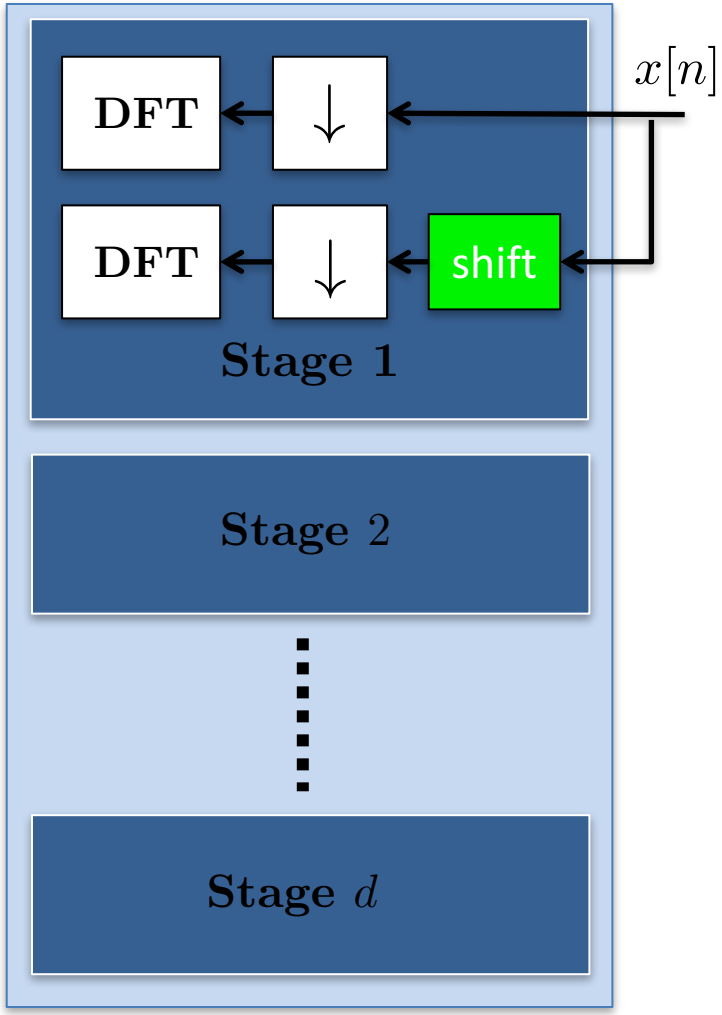
# Main idea


Stage 1
downsample by 5
$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$
single-ton
single-ton
multi-ton
zero-ton
zero-ton
multi-ton
zero-ton
single-ton
multi-ton
Stage 2
downsample by 4

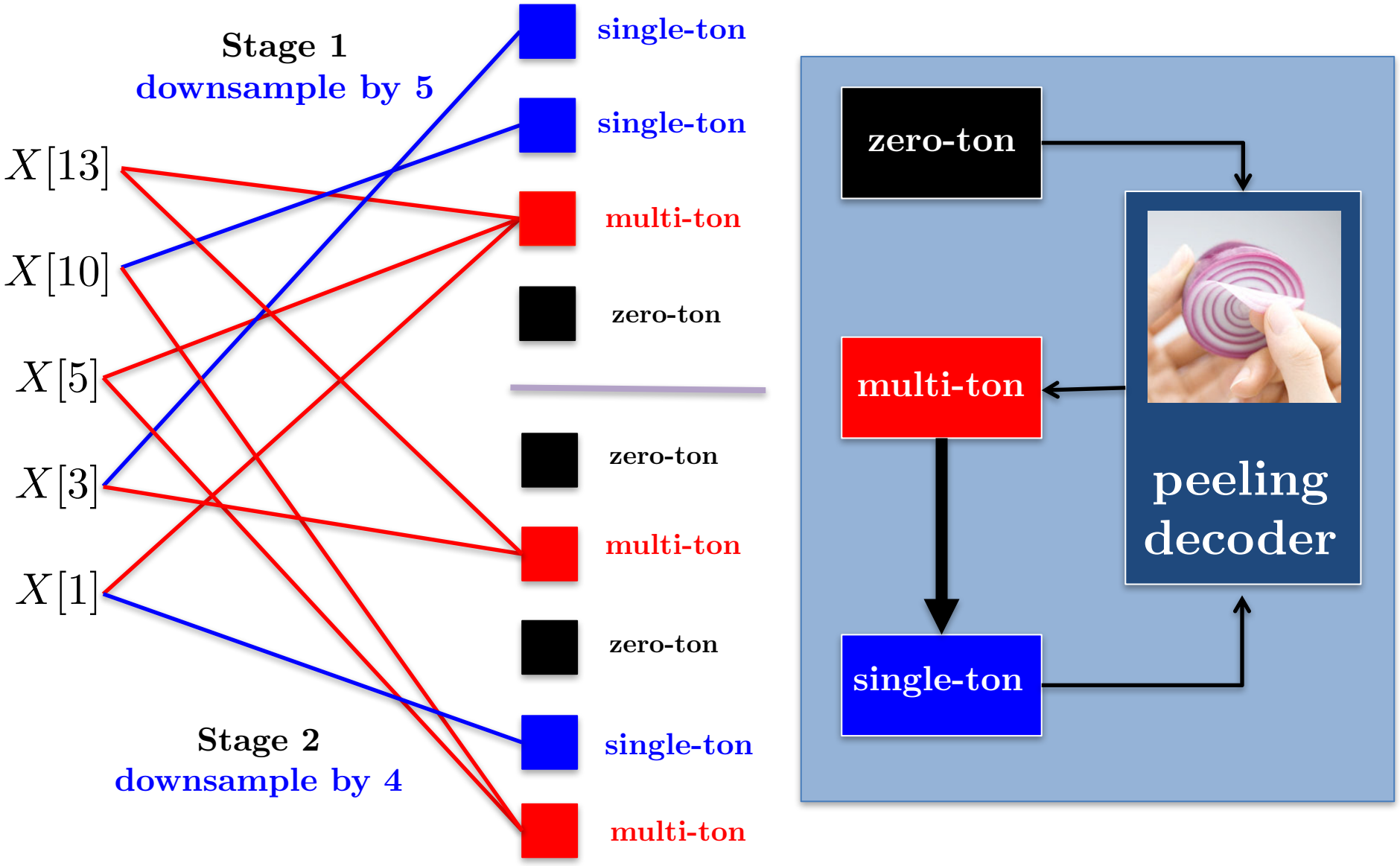

$x[n]$
DFT
↓
DFT
↓
shift
Stage 1
Stage 2

# Main idea

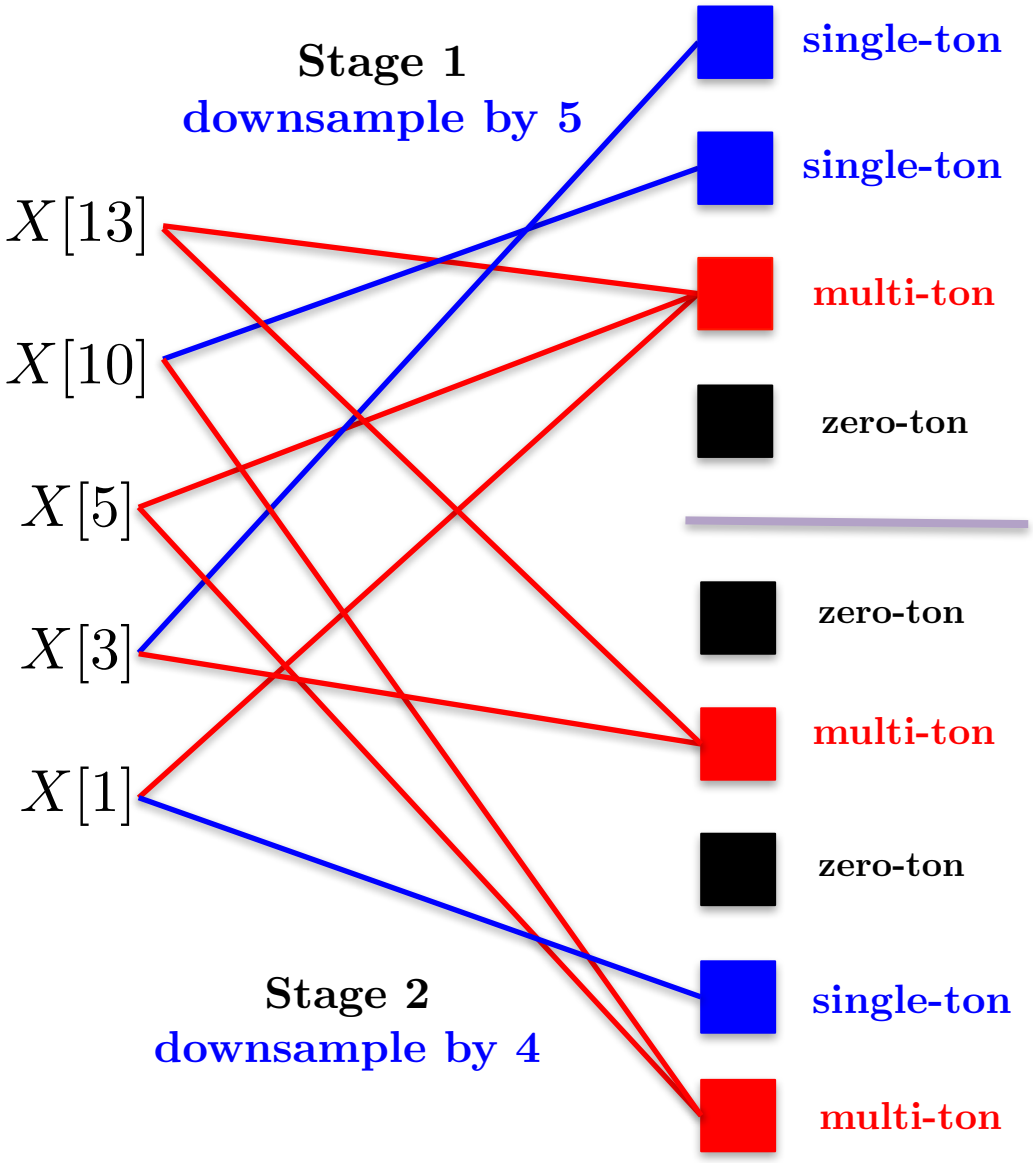

Stage 1
downsample by 5
$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$
single-ton
single-ton
multi-ton
zero-ton
zero-ton
multi-ton
zero-ton
single-ton
multi-ton
Stage 2
downsample by 4


$x[n]$
DFT
↓
DFT
↓
shift
Stage 1
Stage 2
Stage $d$

# Main idea

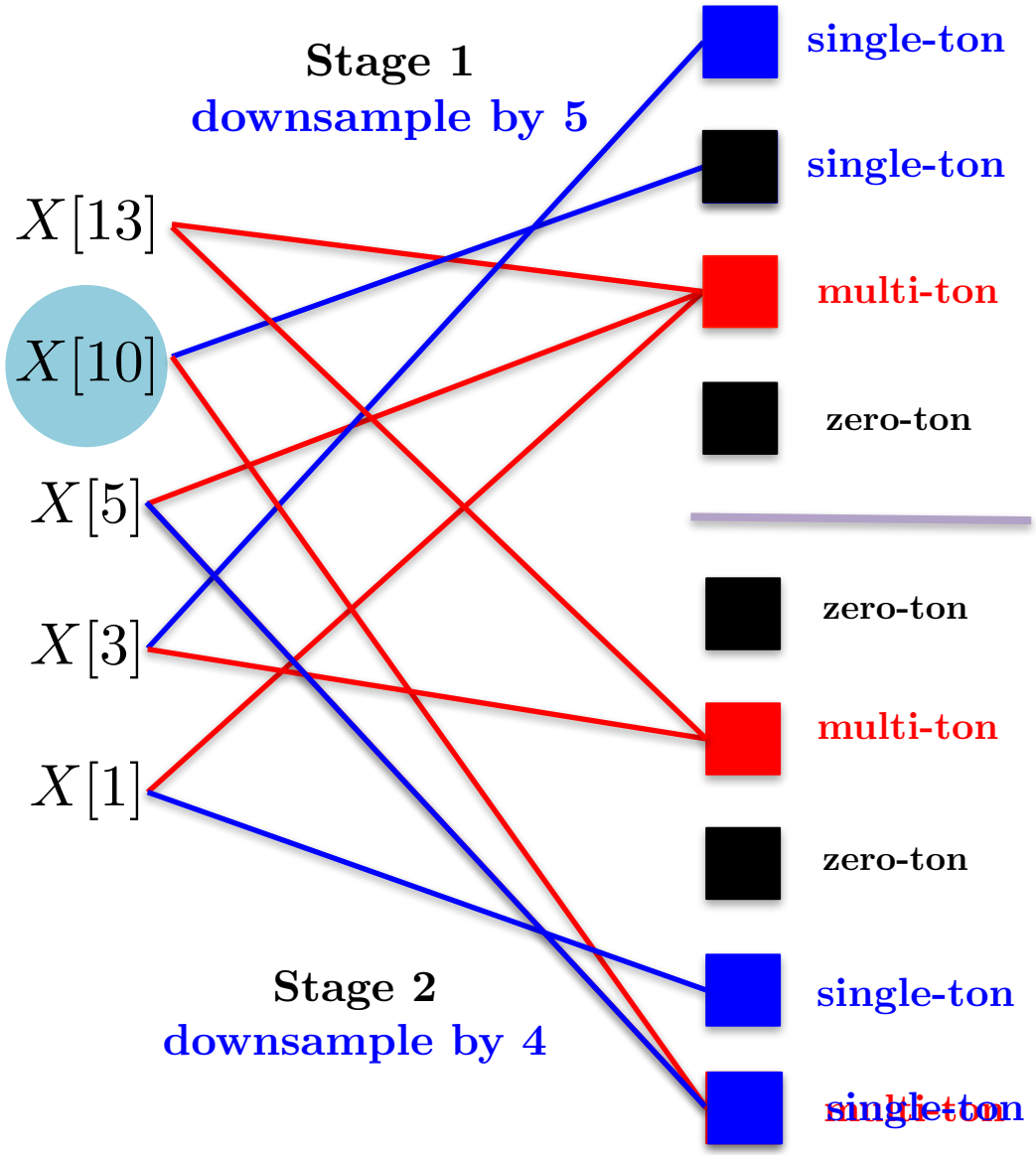

Stage 1
downsample by 5
$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$
Stage 2
downsample by 4
single-ton
single-ton
multi-ton
zero-ton
zero-ton
multi-ton
zero-ton
single-ton
multi-ton
zero-ton
multi-ton
single-ton
peeling decoder

# Main idea


Stage 1
downsample by 5
$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$
Stage 2
downsample by 4
single-ton
single-ton
multi-ton
zero-ton
zero-ton
multi-ton
zero-ton
single-ton
multi-ton

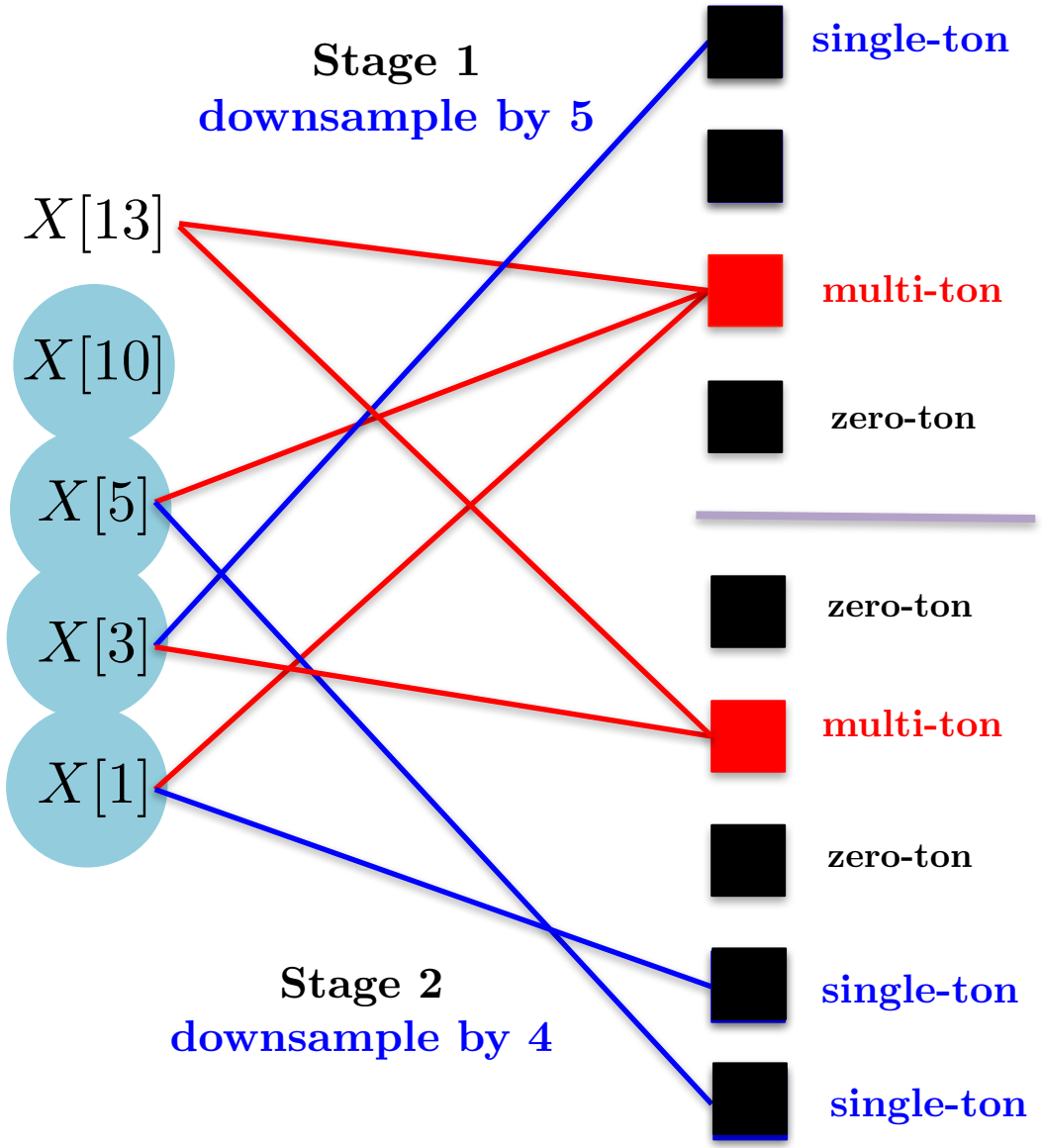

peeling
decoder

# Main idea


Stage 1
downsample by 5
$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$
single-ton
single-ton
multi-ton
zero-ton
zero-ton
multi-ton
zero-ton
single-ton
single-ton
Stage 2
downsample by 4

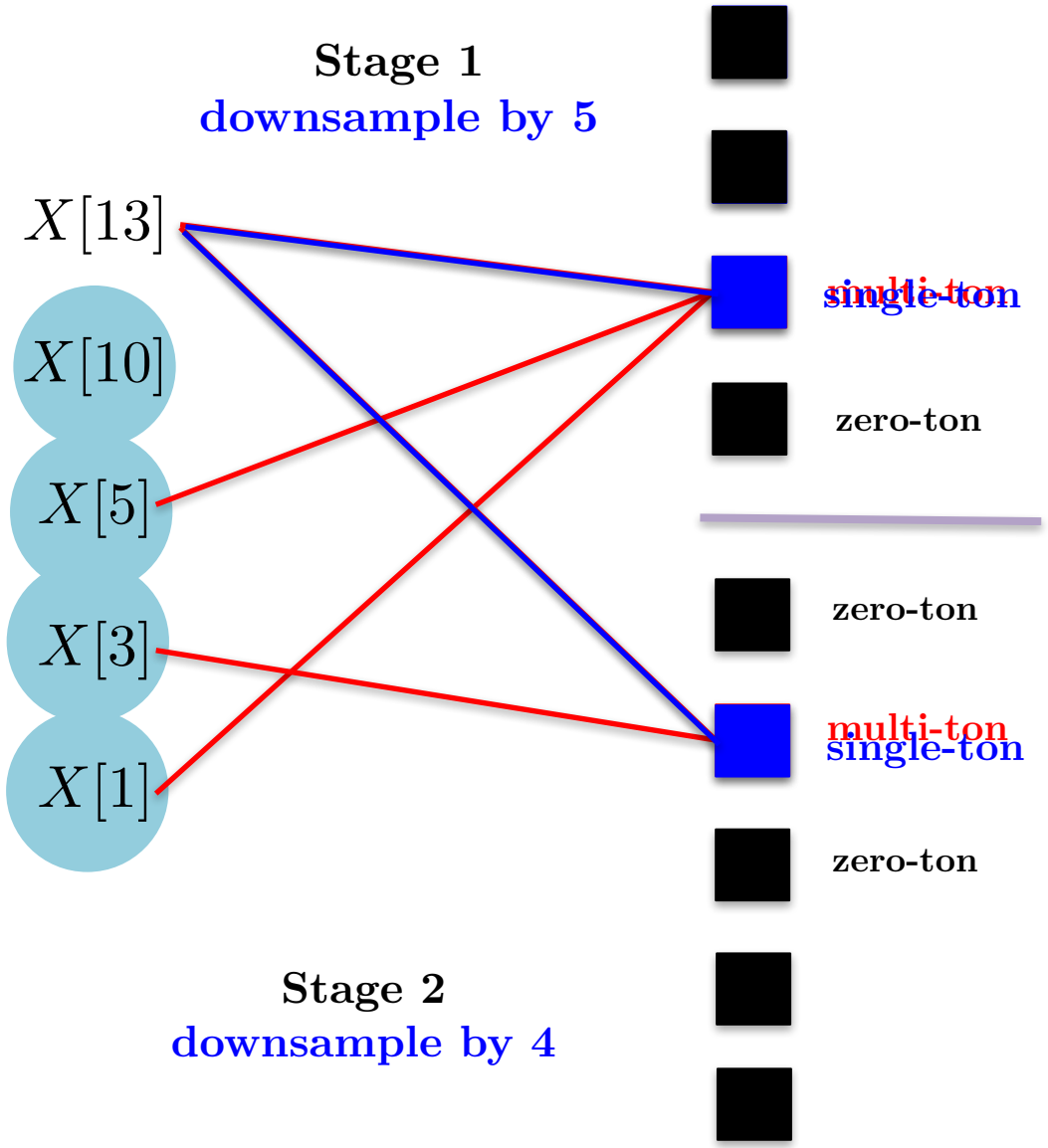

peeling
decoder

# Main idea


Stage 1
downsample by 5
$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$
single-ton
multi-ton
zero-ton
zero-ton
multi-ton
zero-ton
single-ton
single-ton
Stage 2
downsample by 4

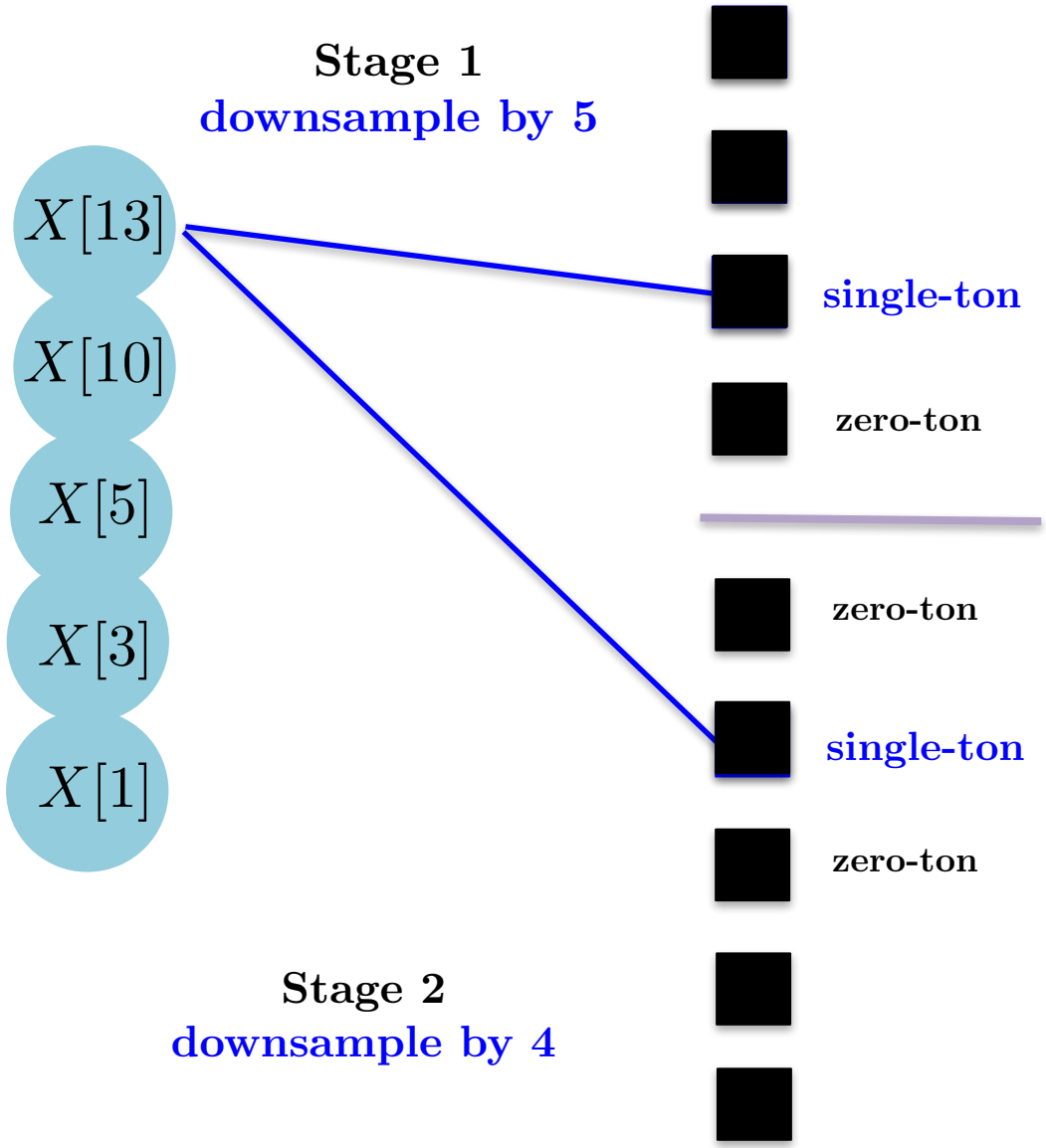

peeling
decoder

# Main idea


Stage 1
downsample by 5
$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$
multi-ton
single-ton
zero-ton
zero-ton
multi-ton
single-ton
zero-ton
Stage 2
downsample by 4


peeling
decoder

# Main idea


Stage 1
downsample by 5
$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$
single-ton
zero-ton
zero-ton
single-ton
zero-ton
Stage 2
downsample by 4


peeling
decoder

# Main idea

**Stage 1**

**downsample by 5**

$X[13]$

$X[10]$

$X[5]$

$X[3]$

$X[1]$

zero-ton

zero-ton

zero-ton

zero-ton

zero-ton

zero-ton

zero-ton

zero-ton

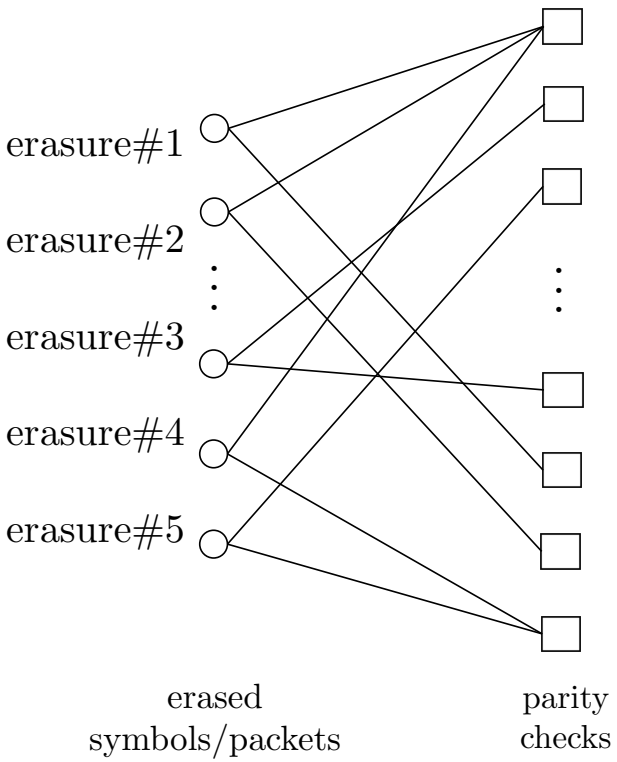
zero-ton

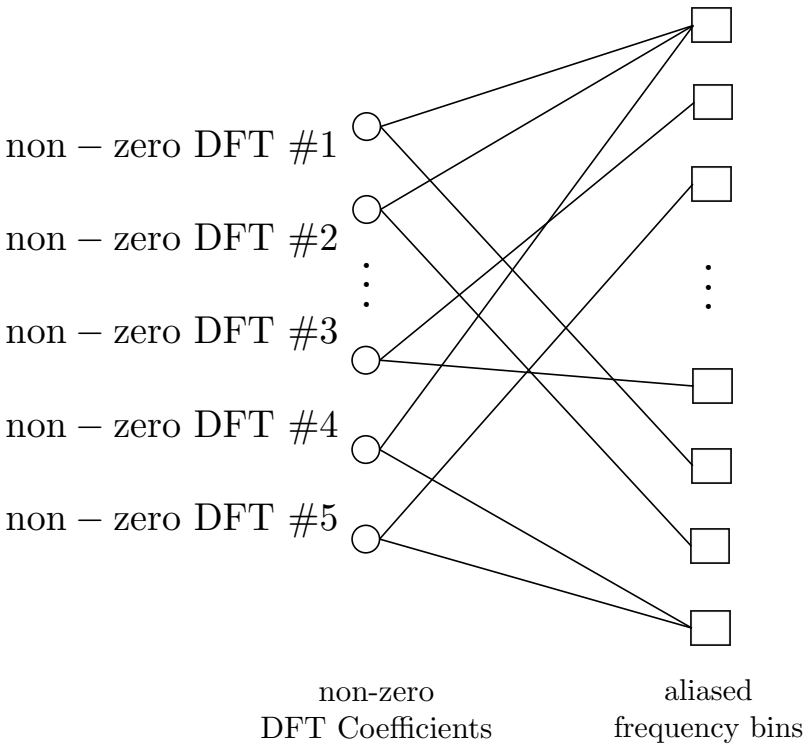
**peeling decoder**

**Sparse DFT Computation**

**= Decoding over Sparse Graphs**

# Sparse DFT Computation = Decoding over Sparse Graphs



# Main Idea

Stage 1
downsample by 5

$X[13]$

$X[10]$

$X[5]$

$X[3]$

$X[1]$

zero-ton

zero-ton

zero-ton

zero-ton

zero-ton

zero-ton

zero-ton

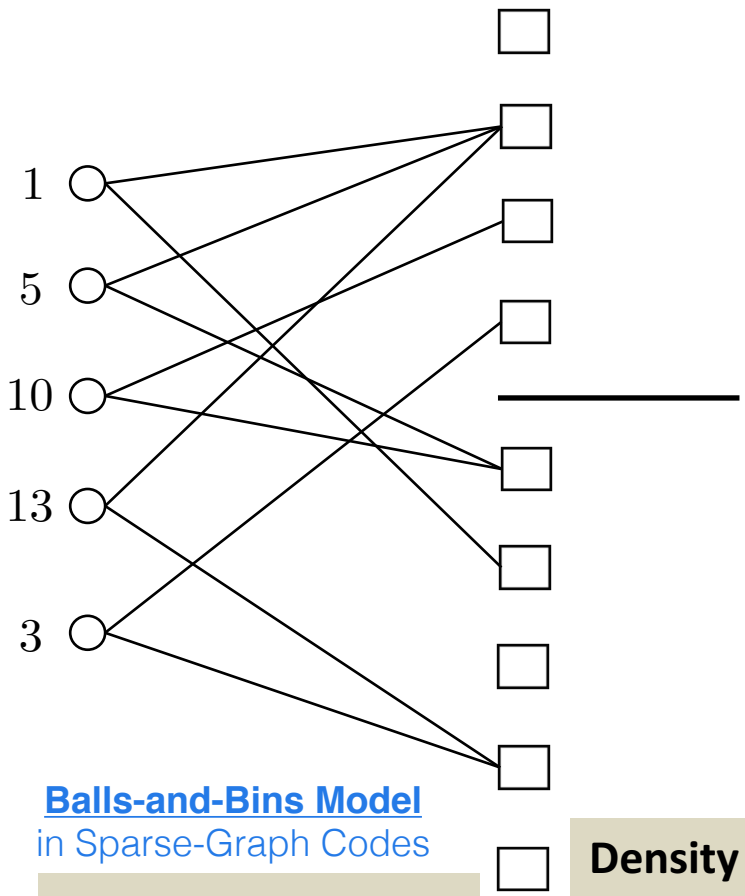
zero-ton

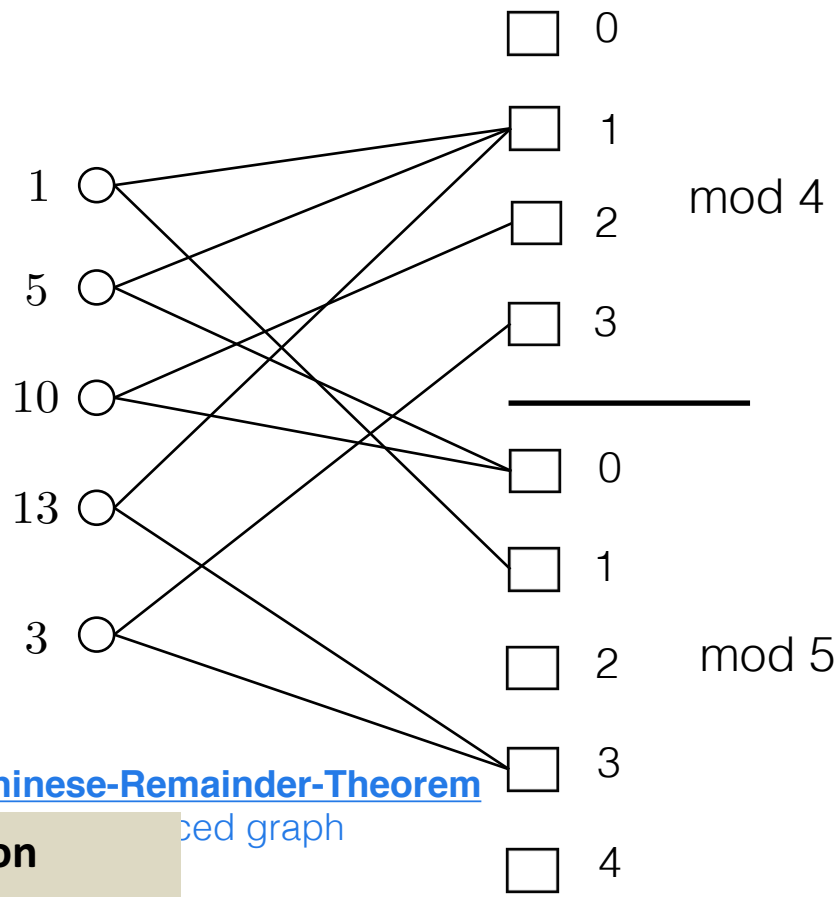
zero-ton




**How do we induce good graphs that will work?**

# CRT-guided Subsampling Induces Good Graphs




**Balls-and-Bins Model**
in Sparse-Graph Codes

**Chinese-Remainder-Theorem**
ced graph

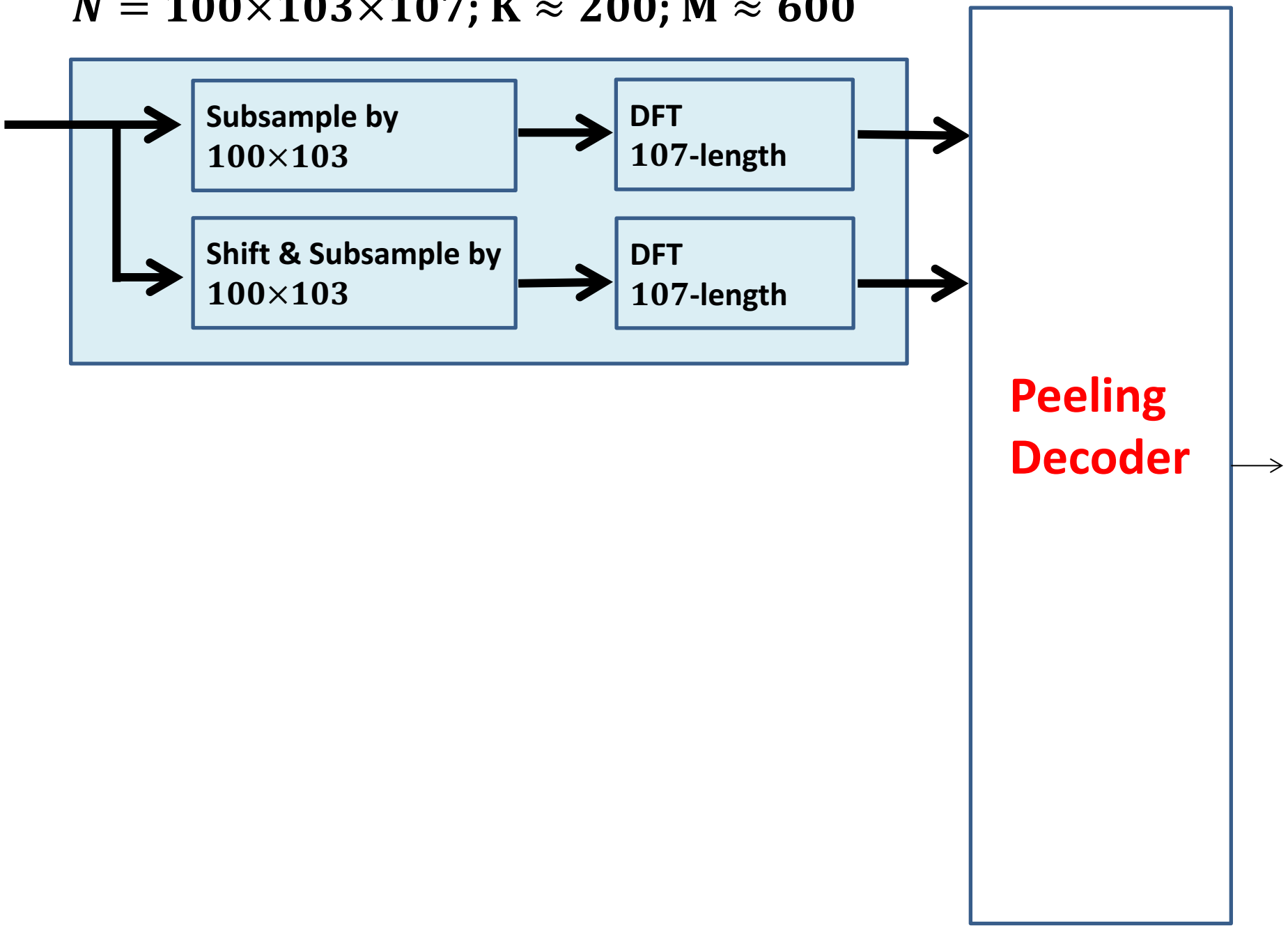
LDPC codes

**Density evolution**
**Martingale**
**Expander graph theory**

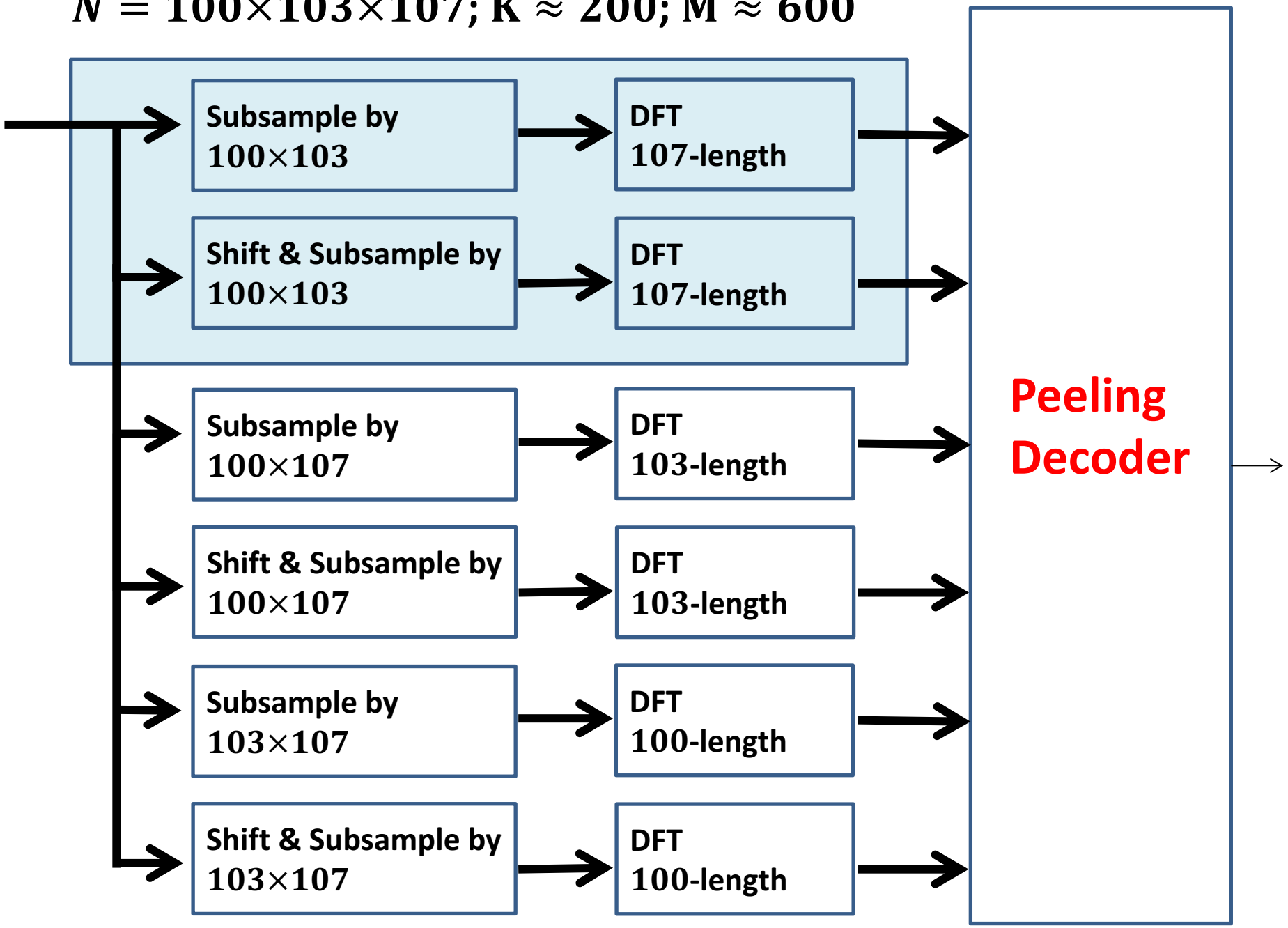
**Chinese-Remainder-Theorem:**
A number between 0-19 is uniquely represented by its remainders modulo (4,5)

> The two graph ensembles are **identical.**

$N = 100 \times 103 \times 107; \mathrm{K} \approx 200; \mathrm{M} \approx 600$



$N = 100 \times 103 \times 107$; $K \approx 200$; $M \approx 600$



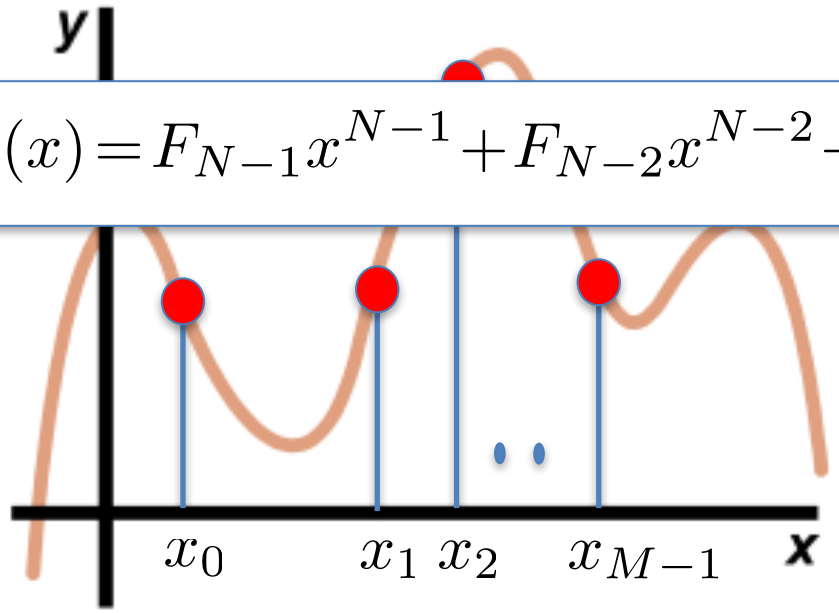
# Sparse polynomial learning

$$x_i \longrightarrow \boxed{f(x)} \longrightarrow f(x_i)$$

**E.g. deg. N=1 million**
**Sparsity K= 200**

$$f(x) = F_{N-1}x^{N-1} + F_{N-2}x^{N-2} + \cdots + F_0$$

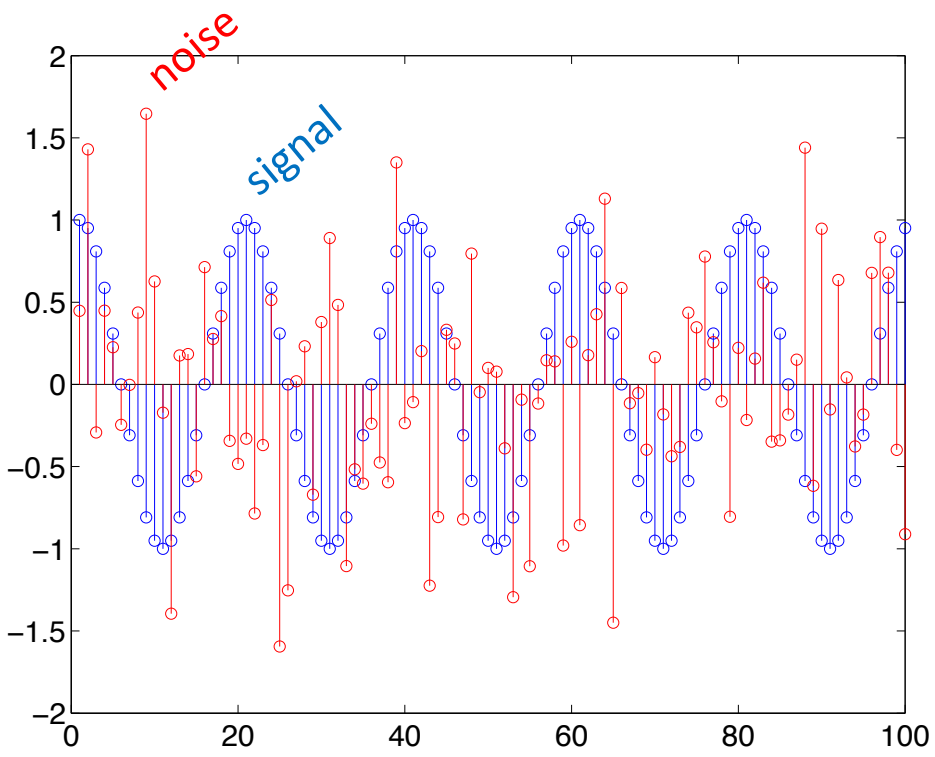
**# evals.** $M = 616$

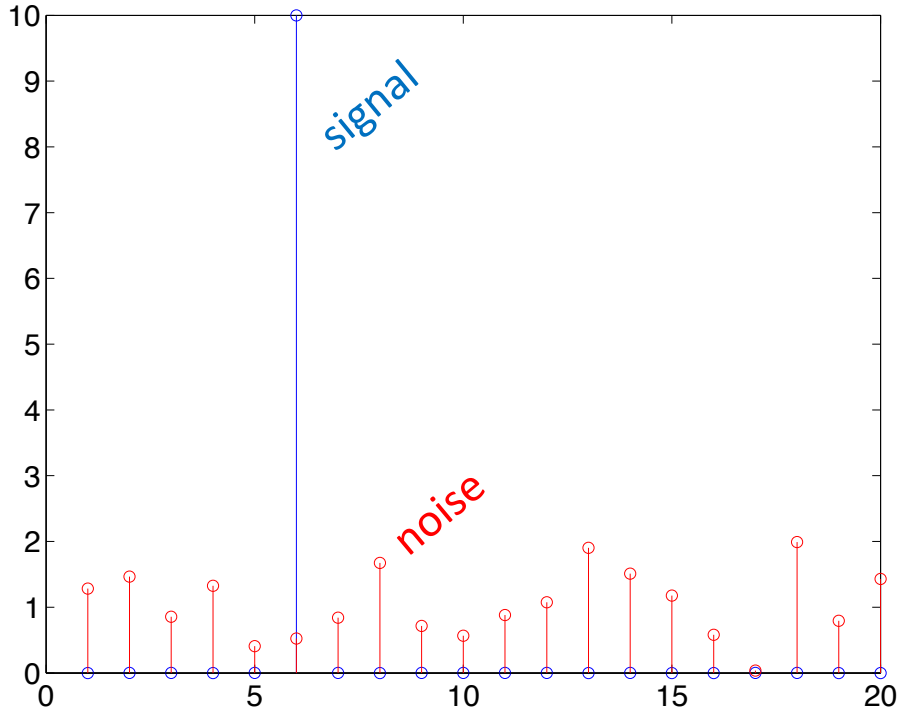



- **N=100x103x107**
- **K≅** $N^{1/3}$
- ***M=2*(100+103+107)-4***

***What if only (very few) K of the N*** **polynomial coeffs.**$\{F_n\}$ **are non-zero?**

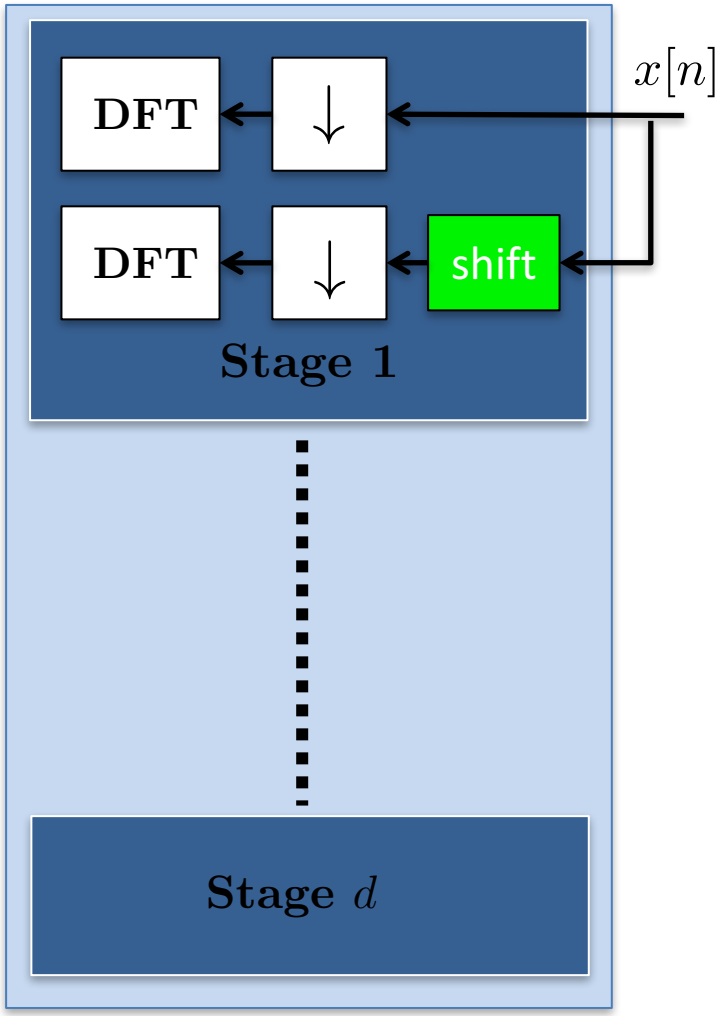
# Noisy Recovery of Sparse DFT



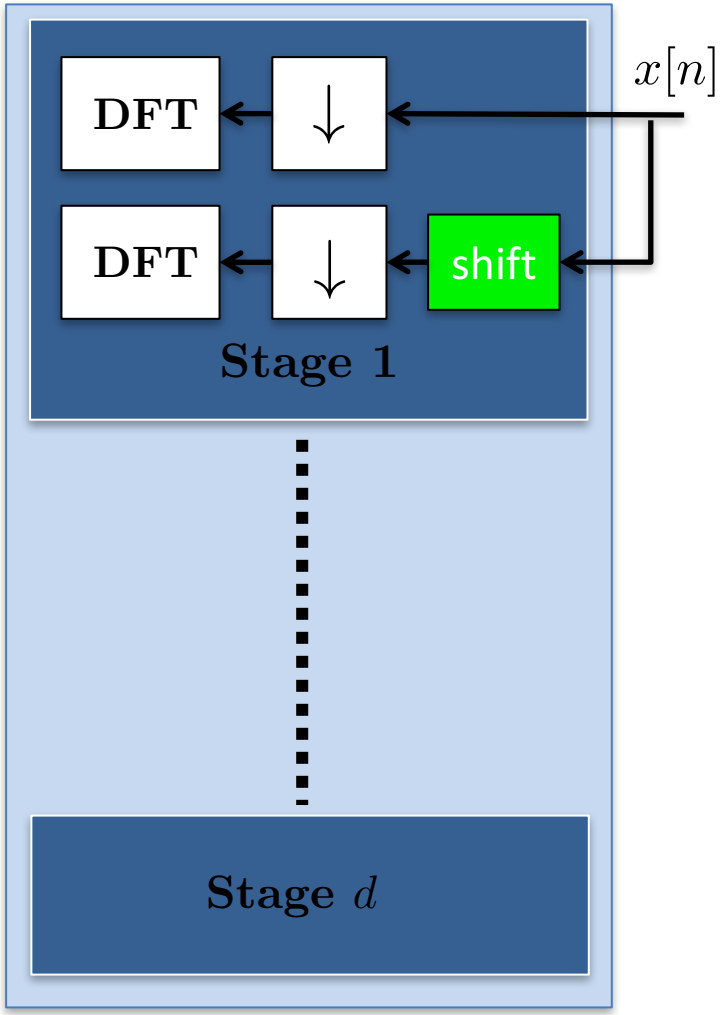
# From Noiseless to Noisy

Noiseless - FFAST

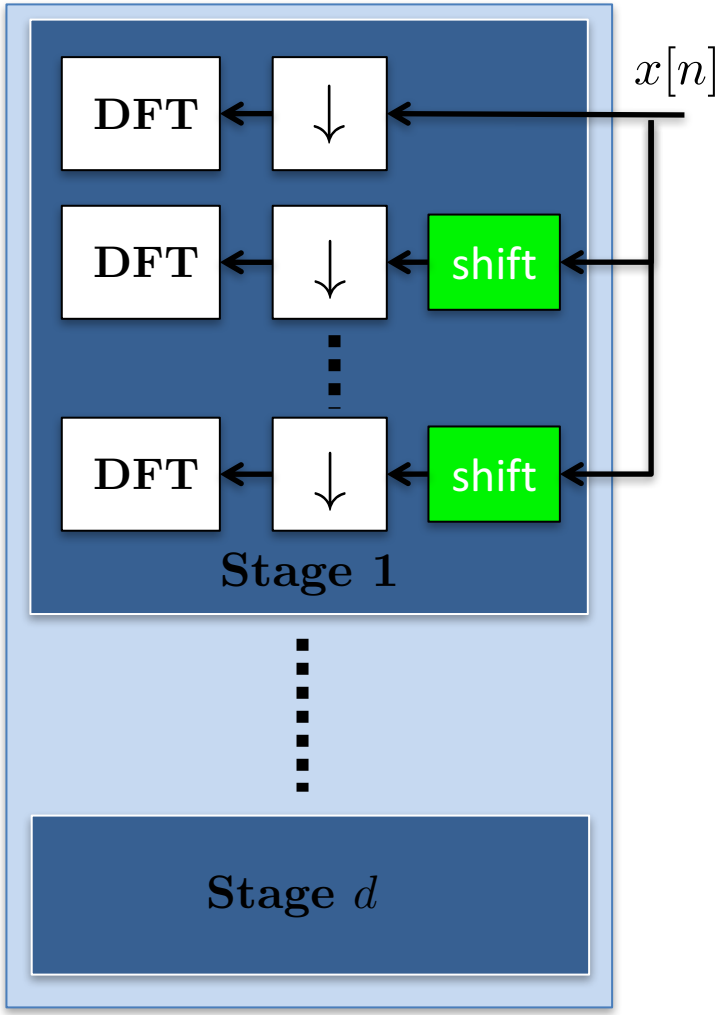


# From Noiseless to Noisy

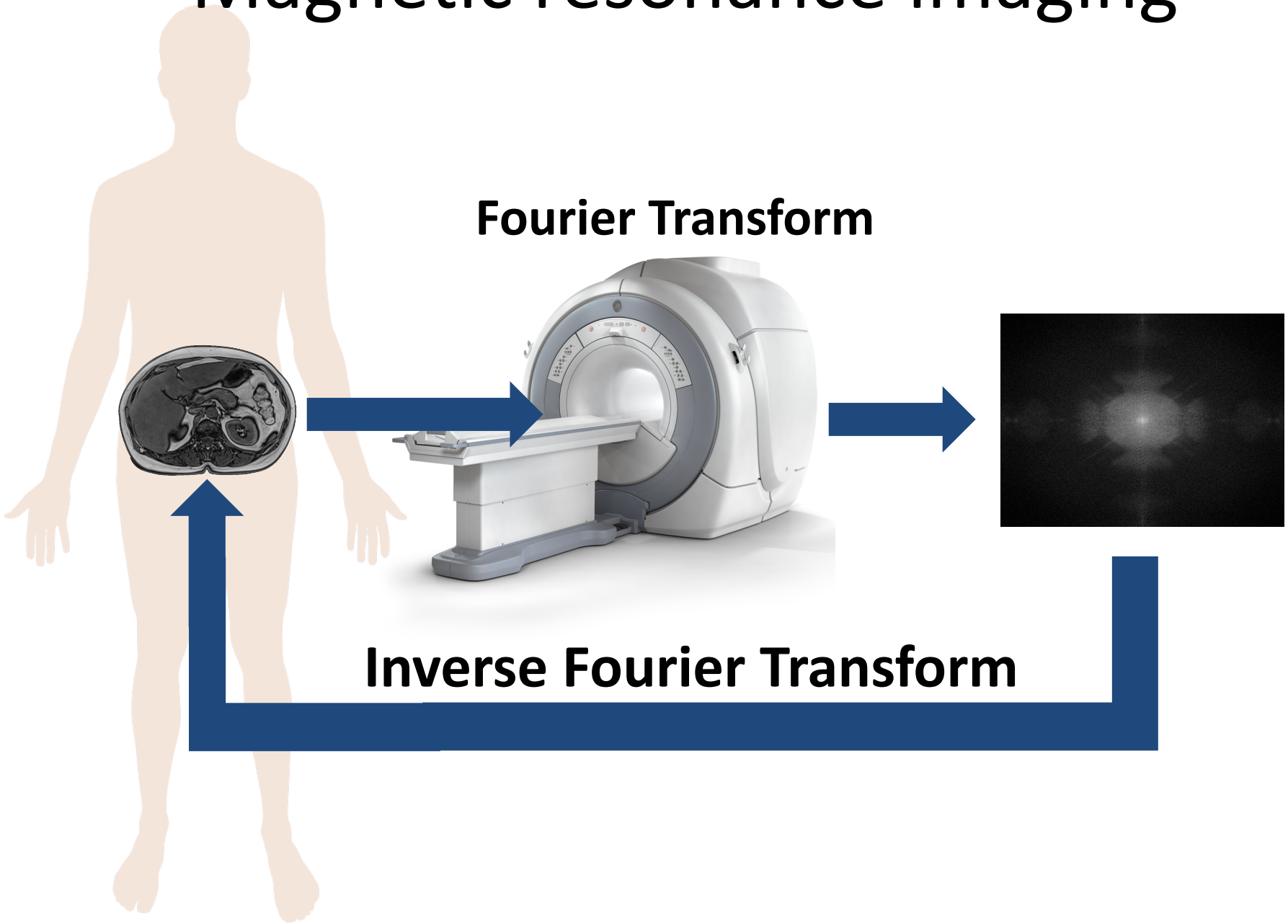
Noiseless - FFAST



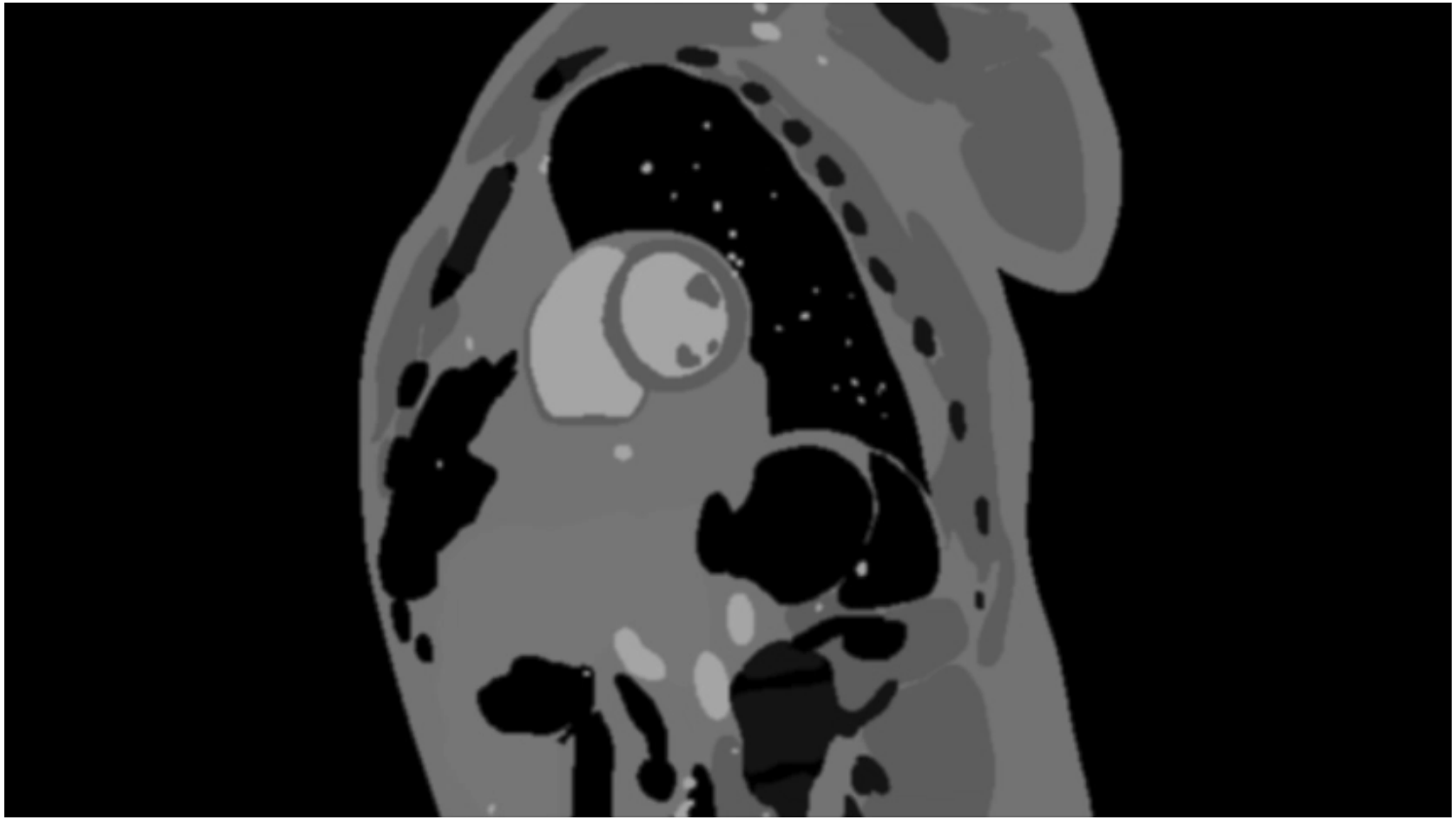

Noisy - R-FFAST



# Magnetic resonance imaging



# Numerical Phantoms for Cardiovascular MR



http://www.biomed.ee.ethz.ch/research/bioimaging/cardiac/mrxcat

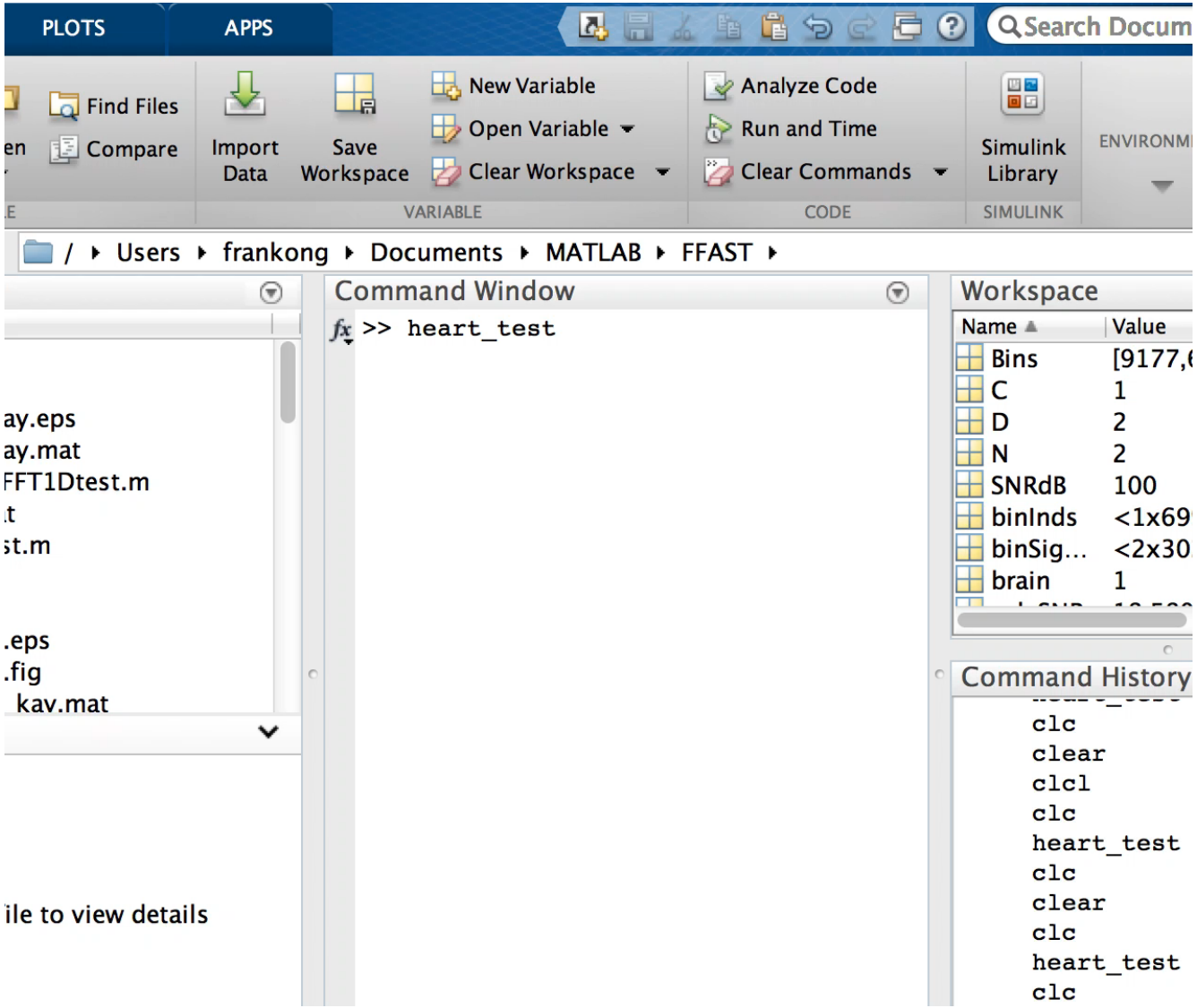
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

$$336 = 16 \times 21$$
$$323 = 17 \times 19$$

# Temporal difference is sparse

**temporal difference** across different frames of the phantom

# Real Time Reconstruction in MATLAB on a Macbook



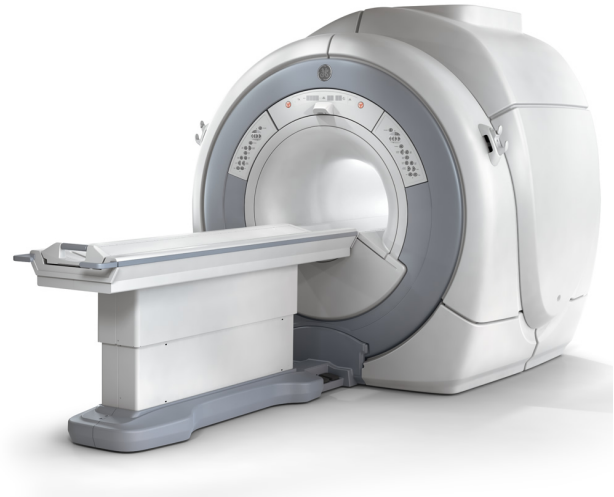

Measurements: 35.33% of Nyquist rate

# MRI Viewfinder

Kodak, 1975


Viewing the photograph


Canon, 2000



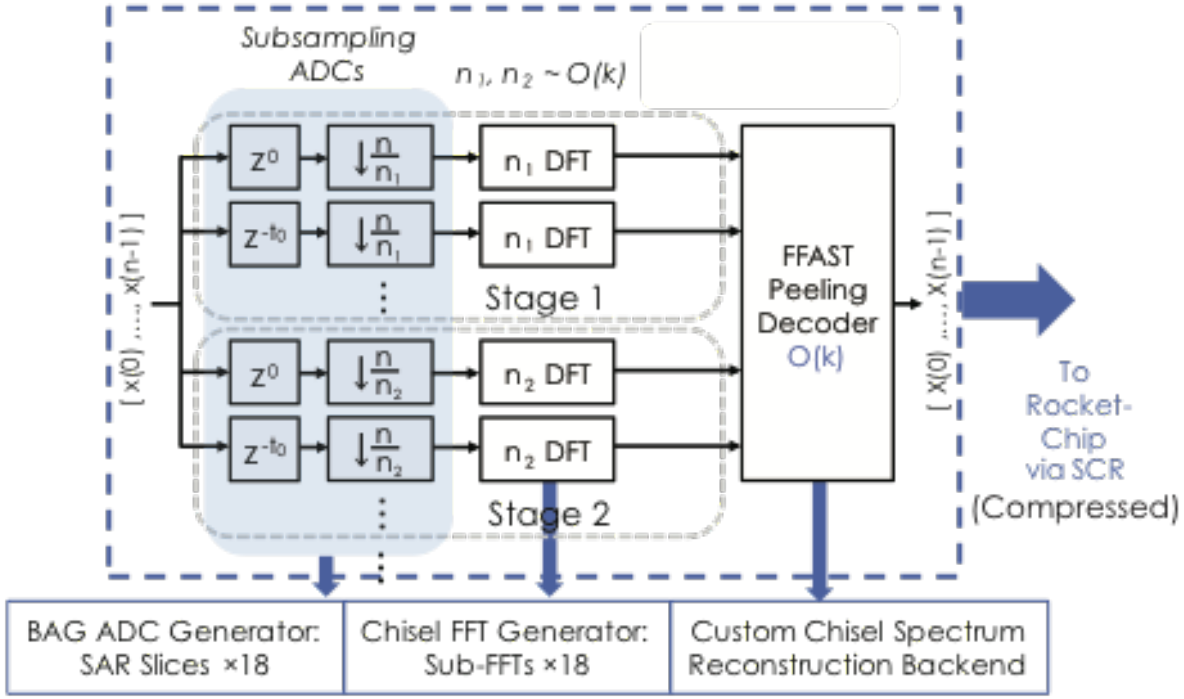
**Real-time MRI with viewfinder**

# Hardware Implementation

***App.: Detection of "heavy hitters" for IoT/spectrum sensing***

A Real-Time, Analog/Digital Co-Designed 1.89-GHz Bandwidth, 175-kHz Resolution Sparse Spectral Analysis RISC-V SoC in 16-nm FinFET

*Wang et al, *ESSIRC 2018*




| | |
|---|---|
| **Technology** | 16nm FinFET |
| **Sensing BW/Resolution** | 2GHz/185kHz |
| **Analysis Time** | ~0.02ms (3.2% sparsity, 4.5%/0.8% false neg. /pos.) |
| **Compression (% Samples)** | 65% @ ADC, 5.3% @ FFAST Output (3.2% sparsity, signal dependent) |

# Walsh-Hadamard Transform

- N-point Discrete Fourier Transform (DFT)

$$f[m] = \frac{1}{N}\sum_{k=0}^{N-1} F[k]e^{\mathrm{i}\frac{2\pi k}{N}m}, \quad m = 0, \cdots, N-1$$

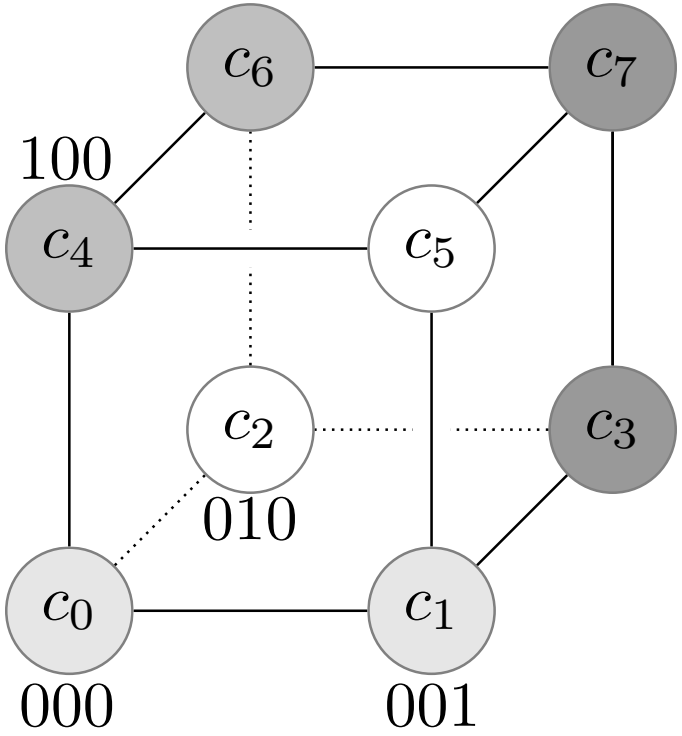
- N-point Walsh-Hadamard (WHT) with $N = 2^n$

$$f[\mathbf{m}] = \sum_{\mathbf{k}\in\{0,1\}^n} F[\mathbf{k}](-1)^{\langle \mathbf{k},\mathbf{m}\rangle}, \quad \mathbf{m} \in \{0,1\}^n$$

- *F*[k] is sparse in many machine learning applications:
  - Decision tree and regression tree
  - Boolean expression (digital logic)
  - Evolutionary biology
  - Hypergraphs

# Walsh-Hadamard Transform

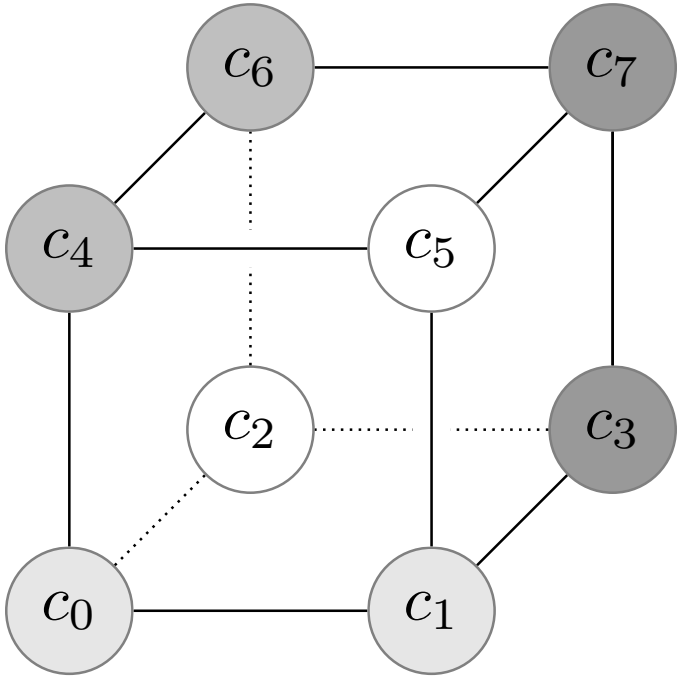
$$\mathbf{f} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{pmatrix}^\top$$



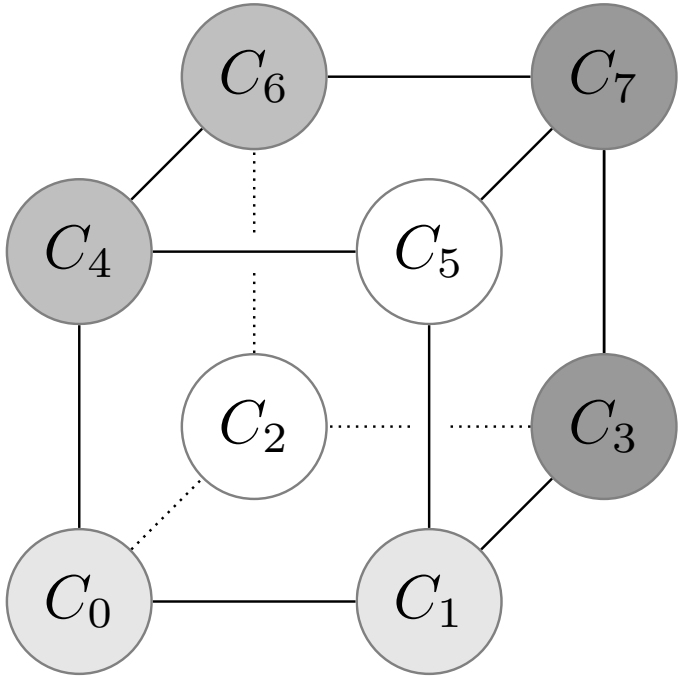

*WHT is equivalent to the high-dimensional DFT over the hyper-cube*

# Walsh-Hadamard Transform

"time" domain

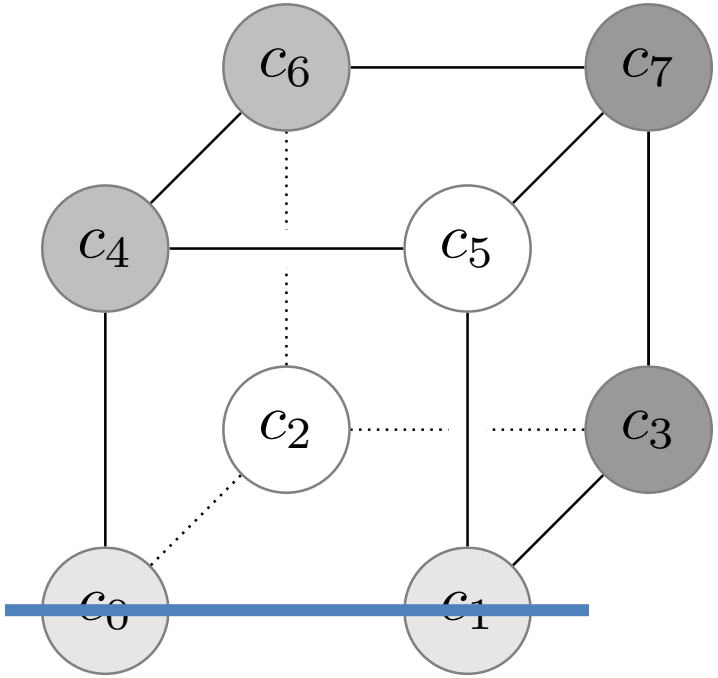



WH domain

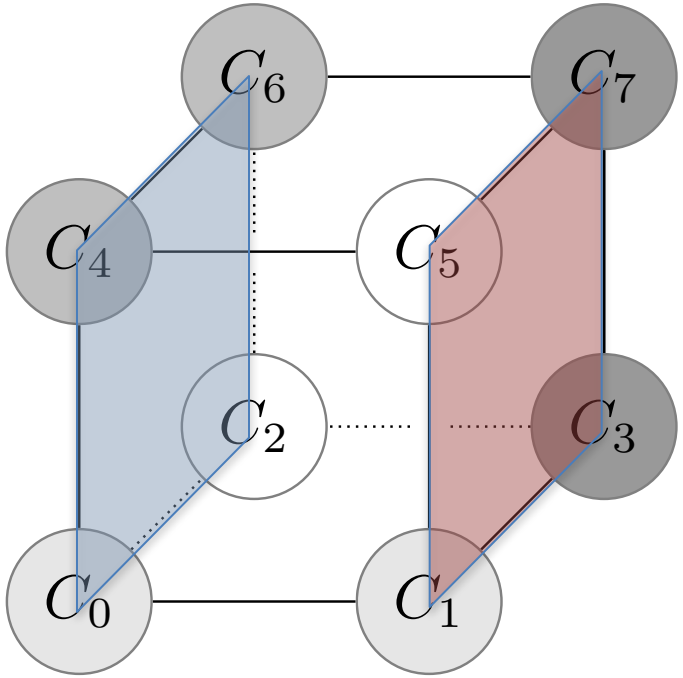


# Walsh-Hadamard Transform

"time" domain

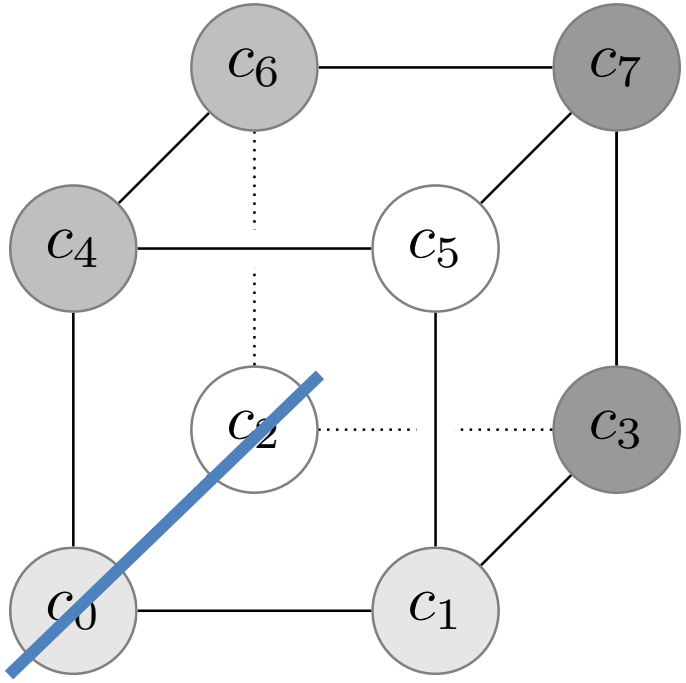



WH domain

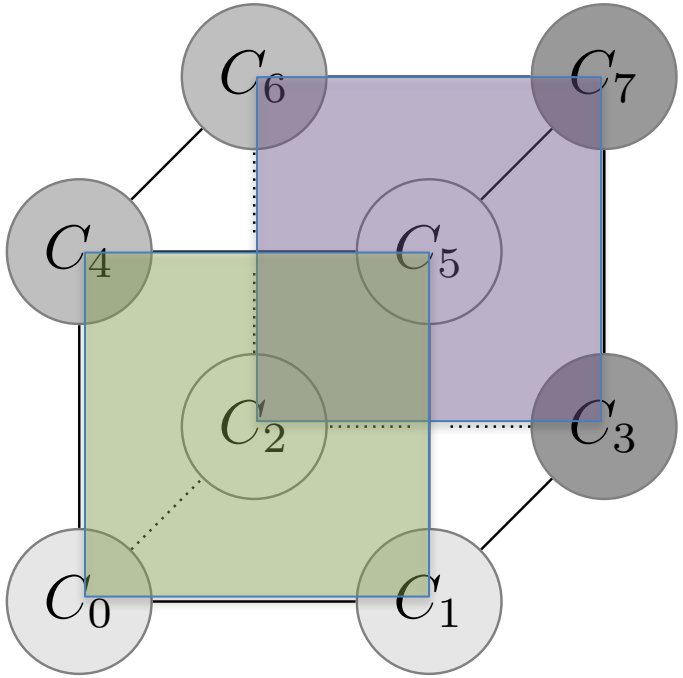


# Walsh-Hadamard Transform

"time" domain

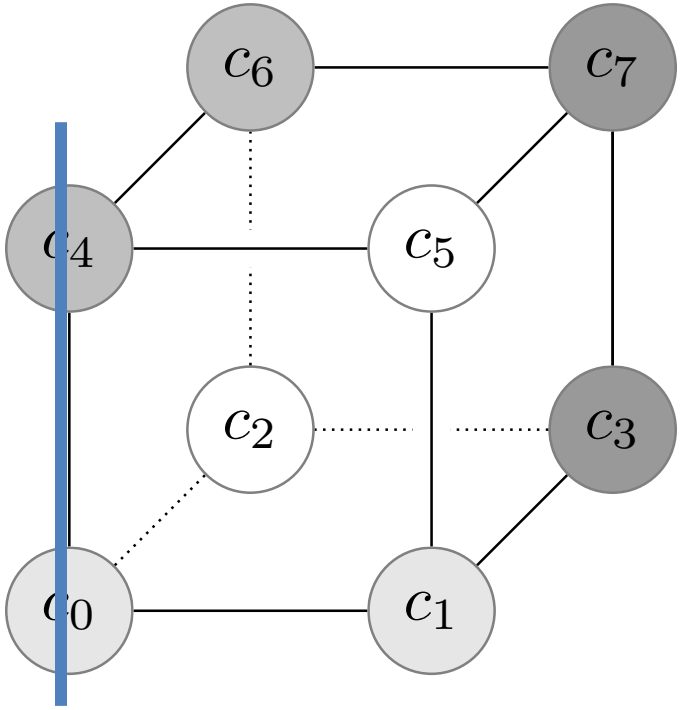



WH domain

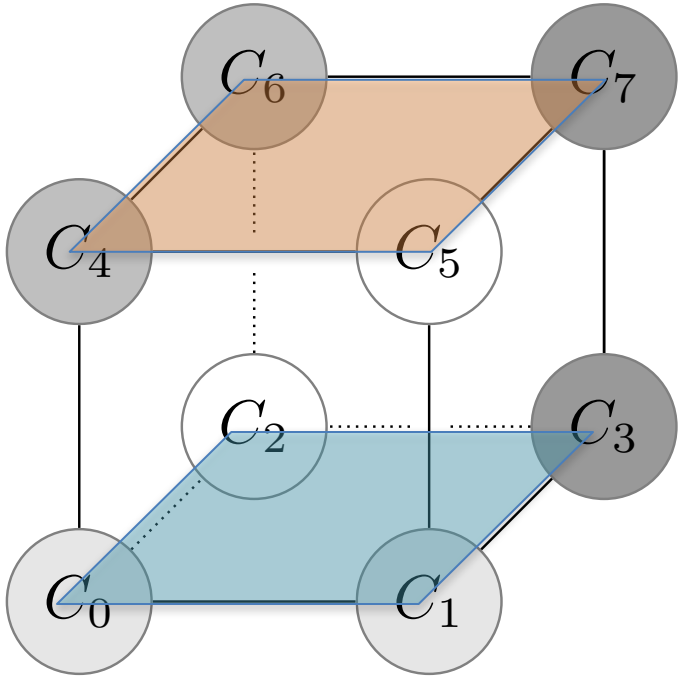


# Walsh-Hadamard Transform

"time" domain




WH domain



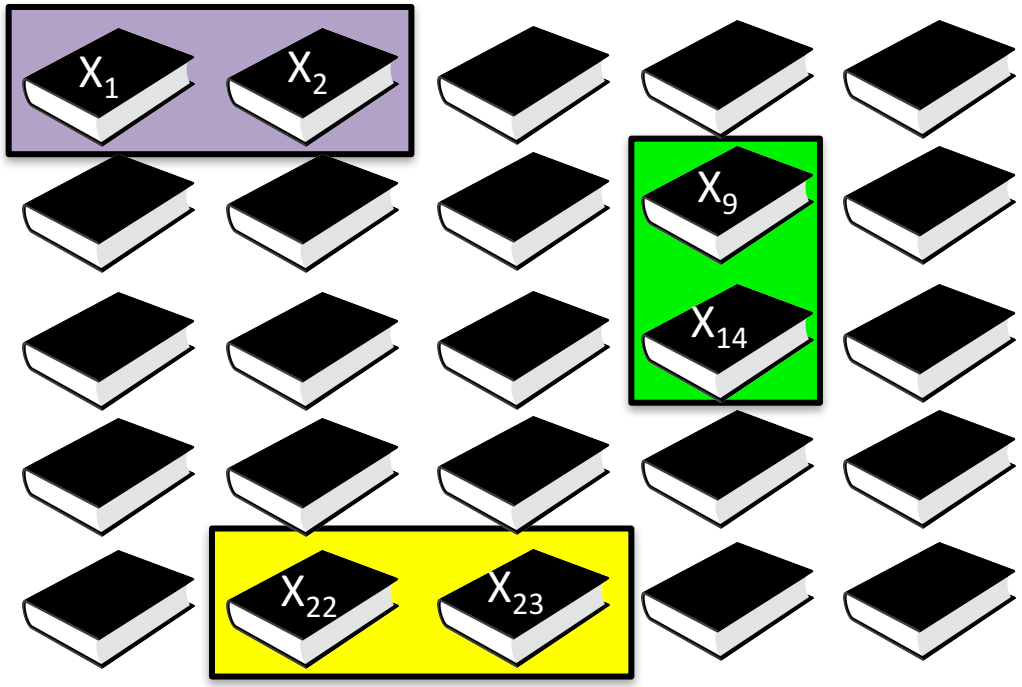
# Walsh-Hadamard Transform

- Connection to polynomials? Let $x_1 = (-1)^{m_1}$ and $x_2 = (-1)^{m_2}$, then

$$f(x_1, x_2) = \sum_{k_1, k_2 \in \{0,1\}} F[k_1, k_2] x_1^{k_1} x_2^{k_2}$$

$$\begin{aligned} f(x_1, x_2) &= 3x_1x_2 + 4x_2, \quad x_1, x_2 \in \{-1, 1\} \\ &= 0 \times x_2^0 x_1^0 + 0 \times x_2^0 x_1^1 + 4 \times x_2^1 x_1^0 + 3 \times x_2^1 x_1^1 \end{aligned}$$

# WHT – Hypergraph Sketching



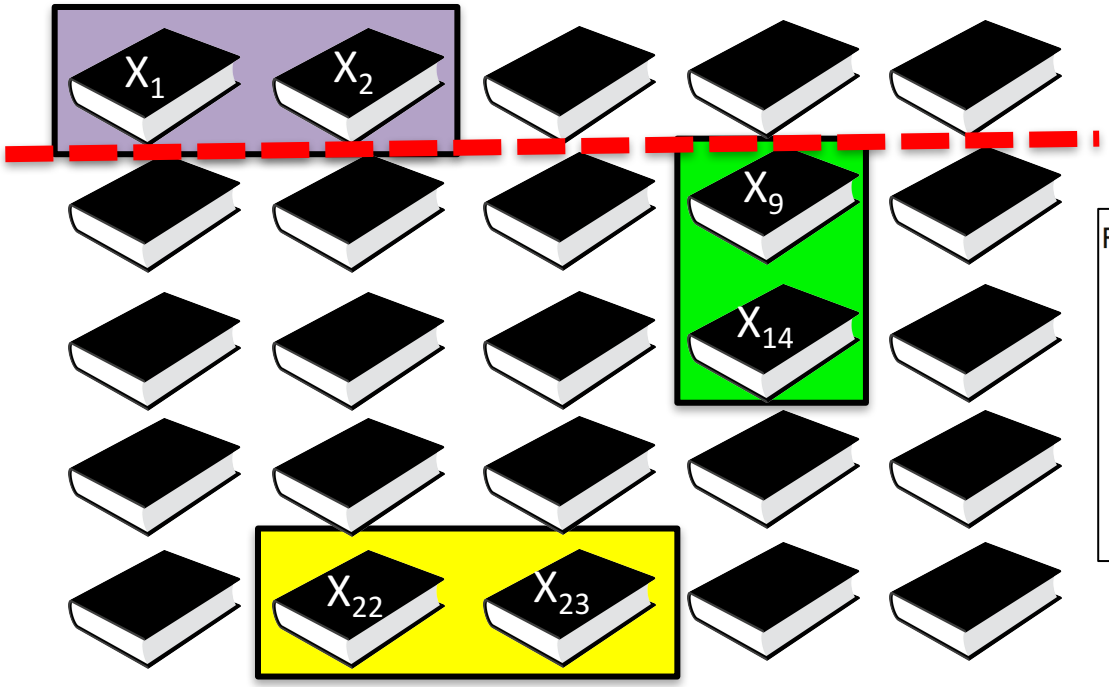

$n = \#$ of books

$s = \#$ of sale patterns


$2^n \approx 10^9$ possible hyperedges if $n = 30$

- recover all sale patterns (hyperedges) without logging every transaction?
- sketch the **cuts** of the graph instead!

# WHT – Hypergraph Sketching




$n = \#$ of books

$s = \#$ of sale patterns


$2^n \approx 10^9$ possible hyperedges if $n = 30$

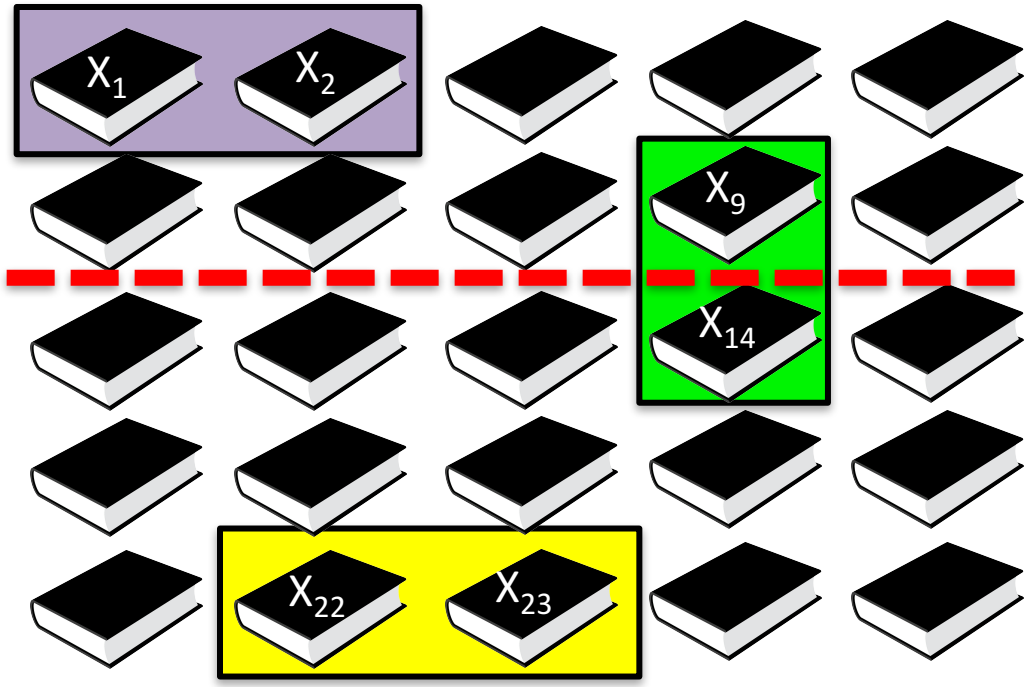
- recover all sale patterns (hyperedges) without logging every transaction?
- sketch the **cuts** of the graph instead!

consider a **cut** :

$$x_1 = \cdots = x_5 = +1$$
$$x_6 = \cdots = x_{25} = -1$$

$\Longrightarrow$ cut value $f(\boldsymbol{x}) = 0$

# WHT – Hypergraph Sketching




$n = \#$ of books

$s = \#$ of sale patterns


$2^n \approx 10^9$ possible hyperedges if $n = 30$

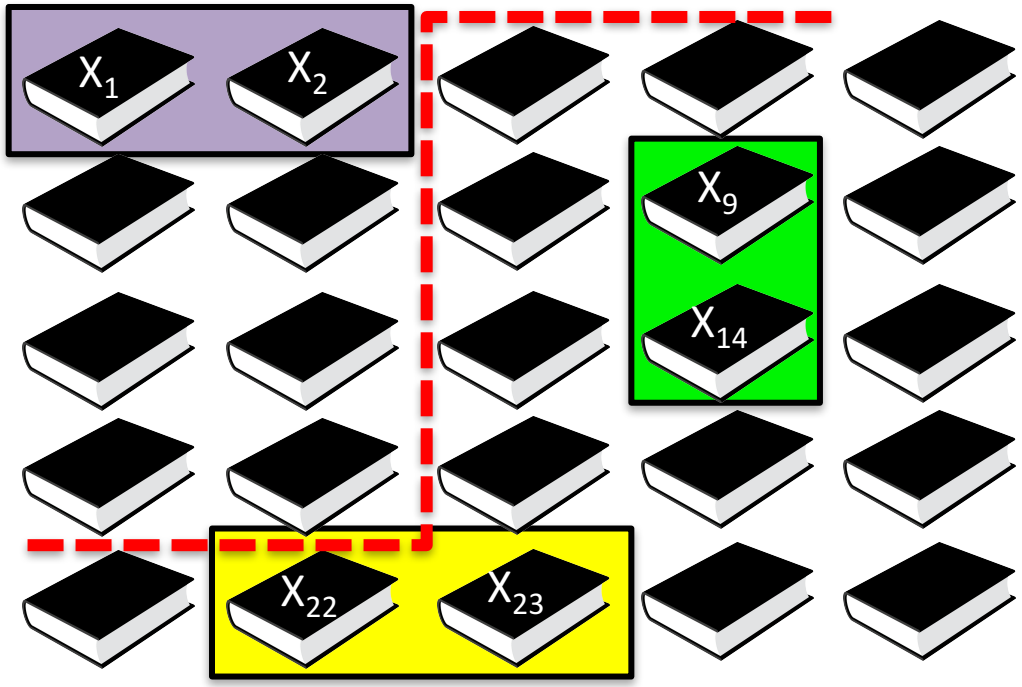
- recover all sale patterns (hyperedges) without logging every transaction?
- sketch the **cuts** of the graph instead!

consider a **cut** :

$$x_1 = \cdots = x_{10} = +1$$
$$x_{11} = \cdots = x_{25} = -1$$

$\Longrightarrow$ cut value $f(\boldsymbol{x}) = 1$

# WHT – Hypergraph Sketching



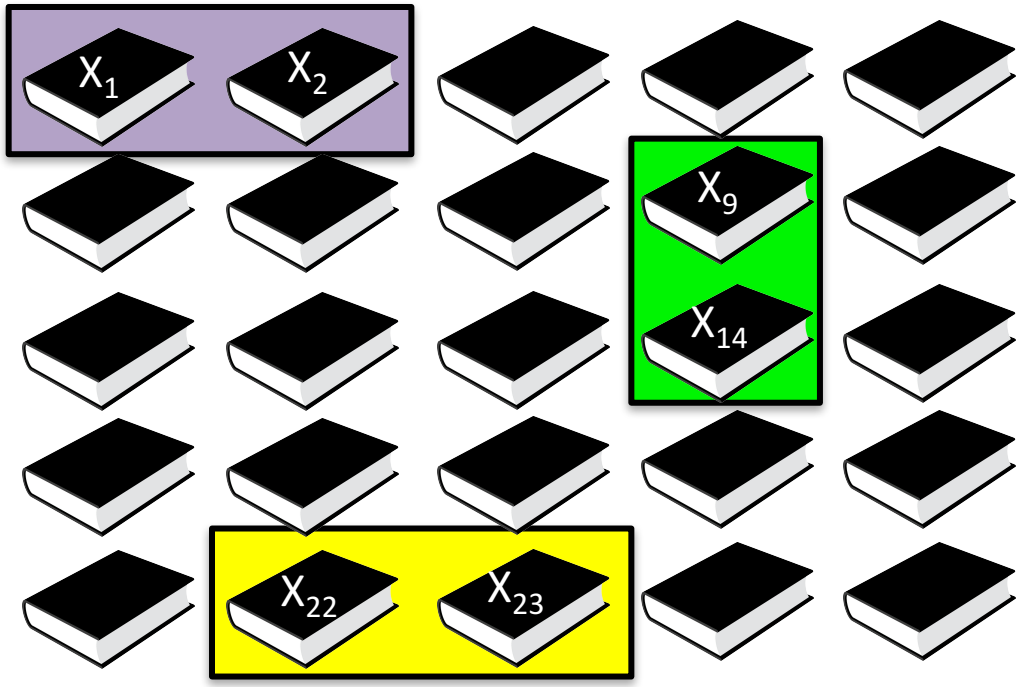

$n = \#$ of books

$s = \#$ of sale patterns


$2^n \approx 10^9$ possible hyperedges if $n = 30$

- recover all sale patterns (hyperedges) without logging every transaction?
- sketch the **cuts** of the graph instead!

# WHT – Hypergraph Sketching




$n = \#$ of books

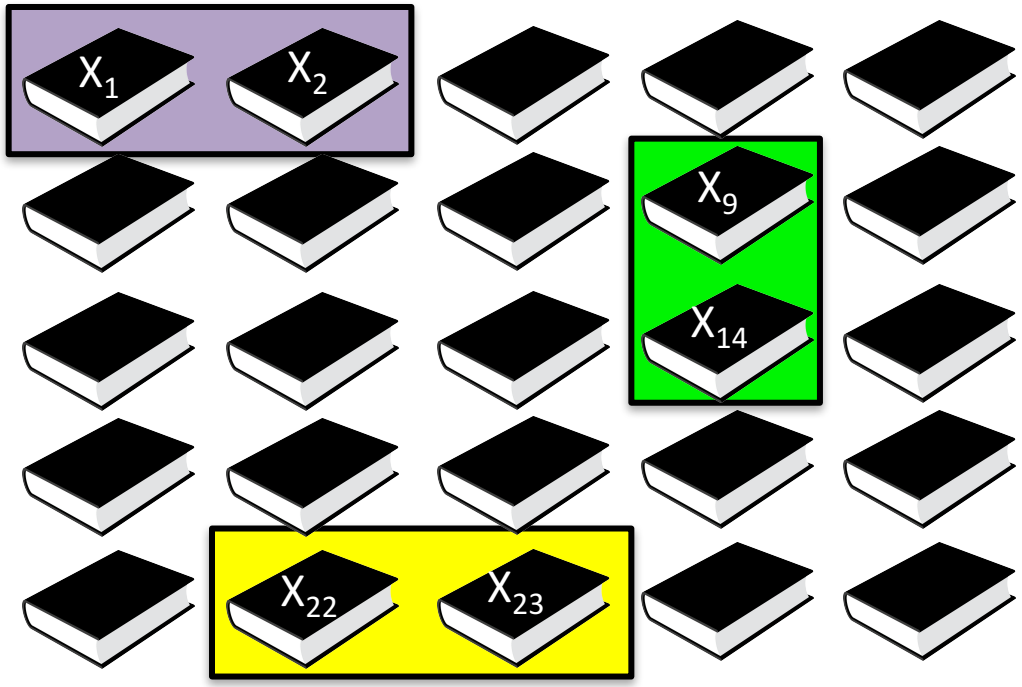
$s = \#$ of sale patterns


$2^n \approx 10^9$ possible hyperedges if $n = 30$

- recover all sale patterns (hyperedges) without logging every transaction?
- sketch the **cuts** of the graph instead!
- Generally speaking, we have the **cut function**

$$f(\boldsymbol{x}) = \frac{3}{2} - \frac{1}{2}x_1x_2 - \frac{1}{2}x_9x_{14} - \frac{1}{2}x_{22}x_{23}$$

# WHT – Hypergraph Sketching




$n = \#$ of books

$s = \#$ of sale patterns


$2^n \approx 10^9$ possible hyperedges if $n = 30$

- small $\#$ of sale patterns $s \ll n$
- small $\#$ of items per sale $d \ll n$

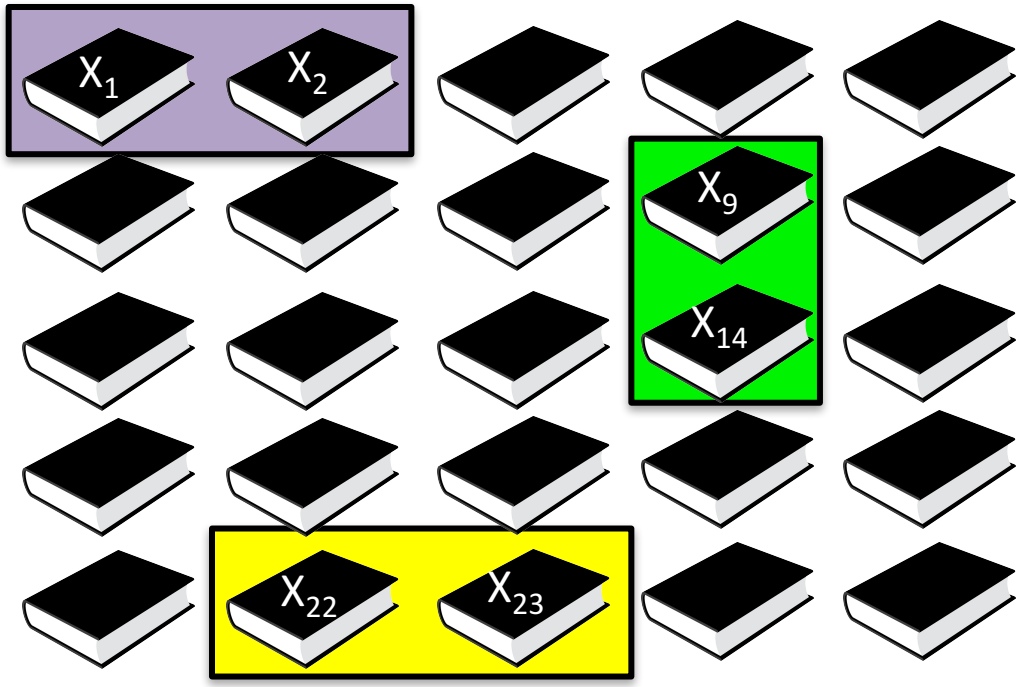
$\Rightarrow$

- $K$-sparse polynomial
- $K \leq s2^{d-1}$

- Generally speaking, we have the **cut function**

$$f(\boldsymbol{x}) = \frac{3}{2} - \frac{1}{2}x_1x_2 - \frac{1}{2}x_9x_{14} - \frac{1}{2}x_{22}x_{23}$$

# WHT – Hypergraph Sketching




$n = \#$ of books

$s = \#$ of sale patterns

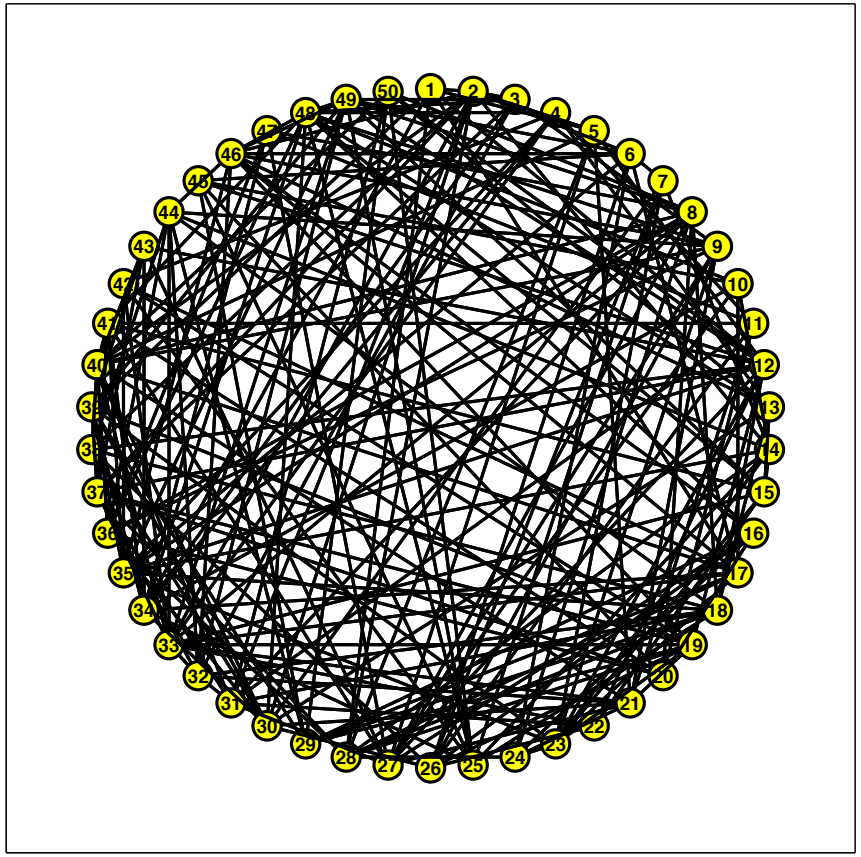

$2^n \approx 10^9$ possible hyperedges if $n = 30$

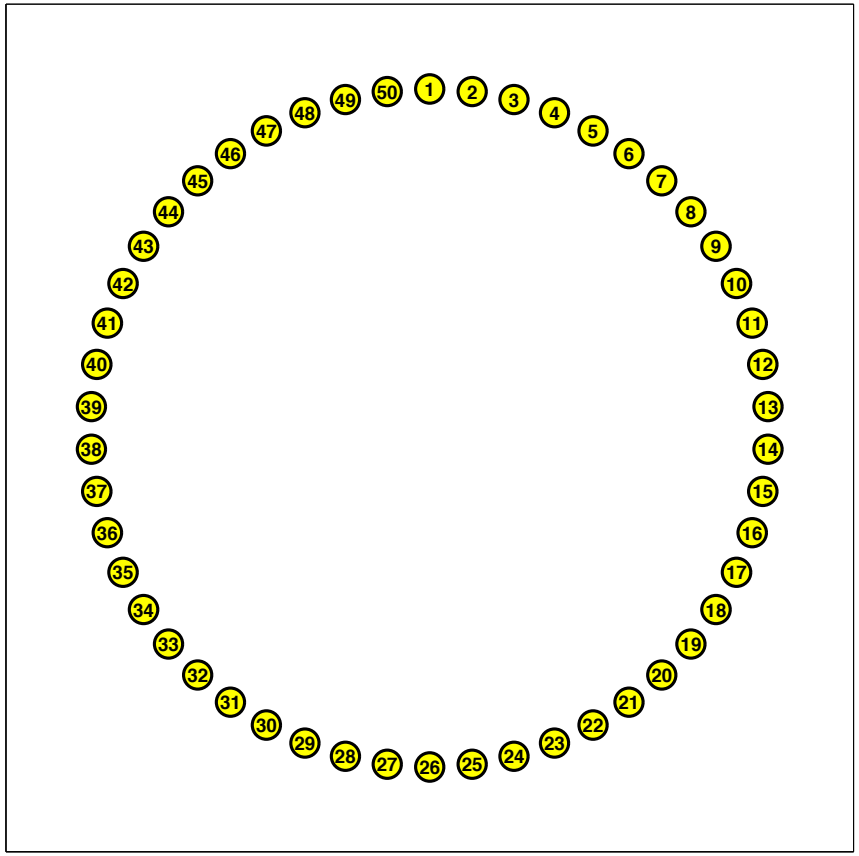
sub-sample **cuts** → recover **hyperedges**

- Generally speaking, we have the **cut function**

$$f(\boldsymbol{x}) = \frac{3}{2} - \frac{1}{2}x_1x_2 - \frac{1}{2}x_9x_{14} - \frac{1}{2}x_{22}x_{23}$$

# WHT – Hypergraph Sketching





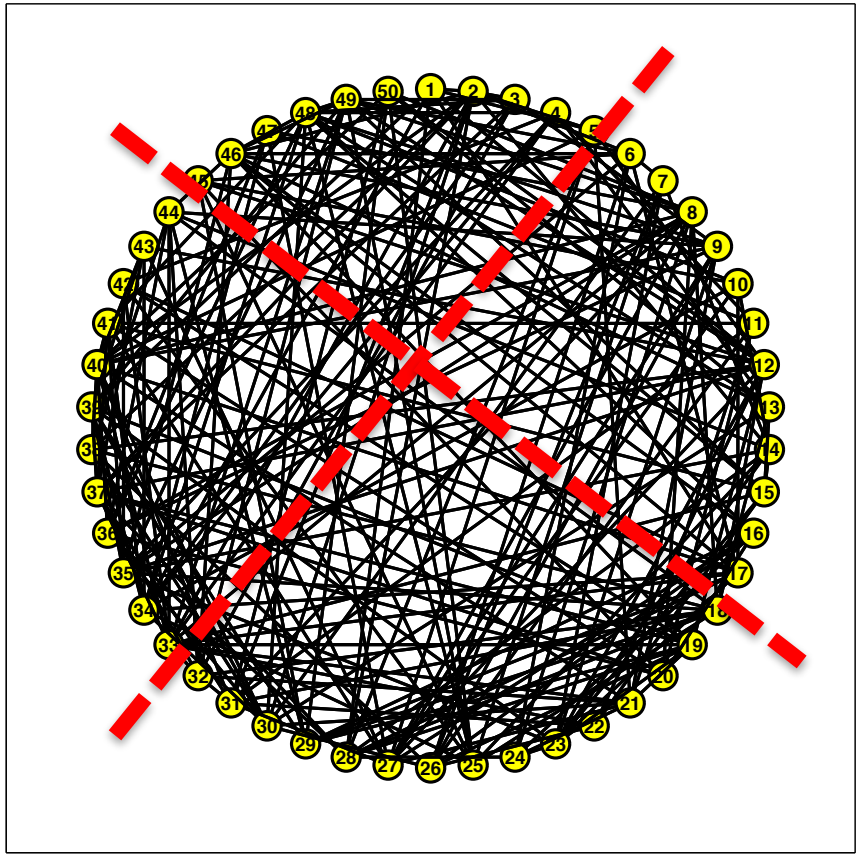
$n = 50$ books

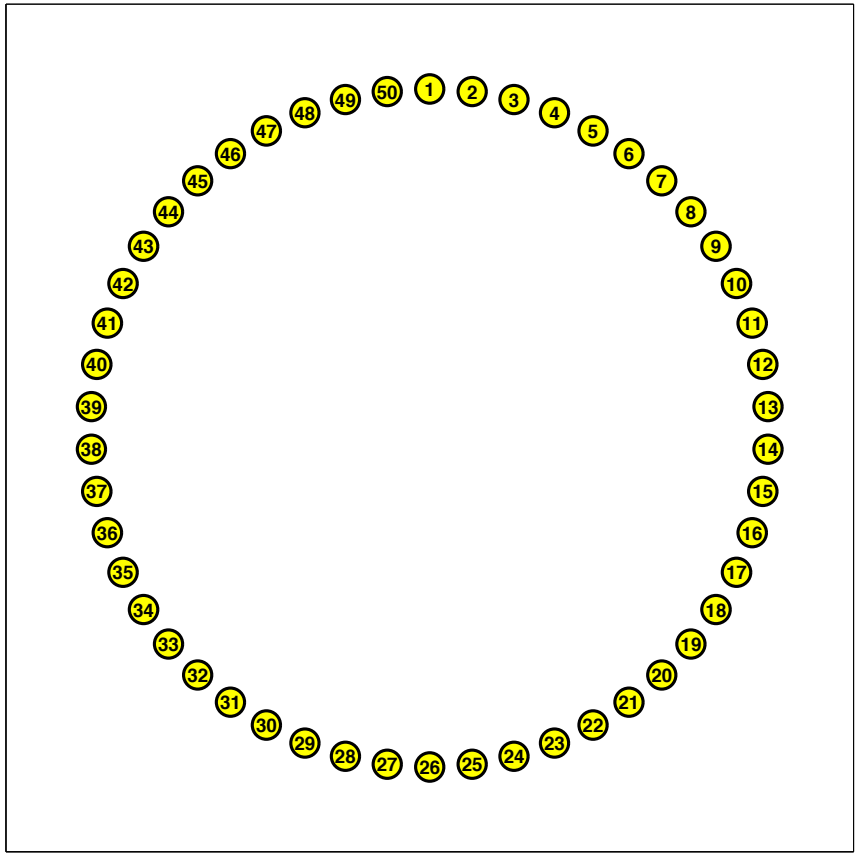
$d = 2$ items/sale

$s = 250$ sale patterns

- total cut values $2^n = 2^{50}$
- sparsity $K \leq s2^{d-1} = 500$
- # of cut queries $O(Kn) \approx 25000$

# WHT – Hypergraph Sketching





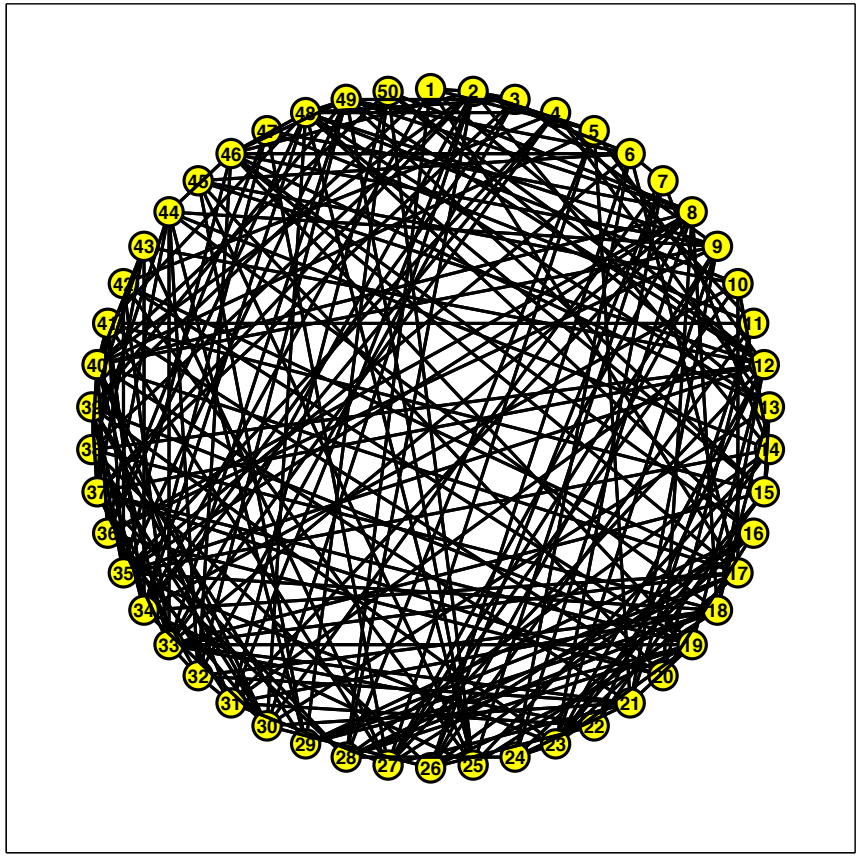
$n = 50$ books

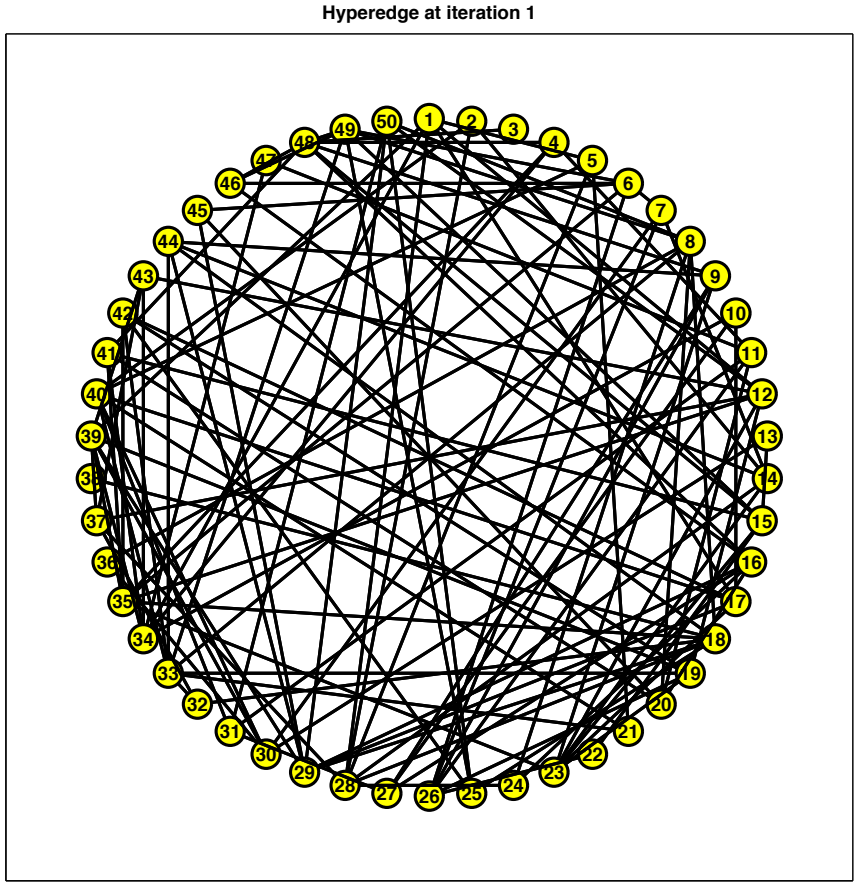
$d = 2$ items/sale

$s = 250$ sale patterns

- total cut values $2^n = 2^{50}$
- sparsity $K \leq s2^{d-1} = 500$
- # of cut queries $O(Kn) \approx 25000$

# WHT – Hypergraph Sketching





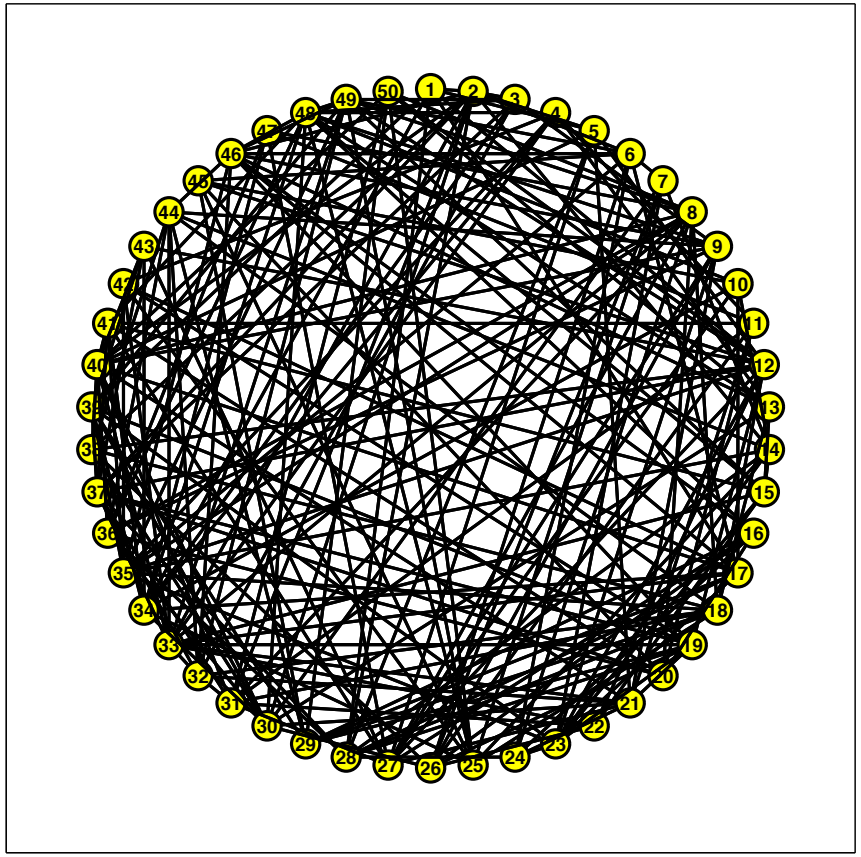

$n = 50$ books

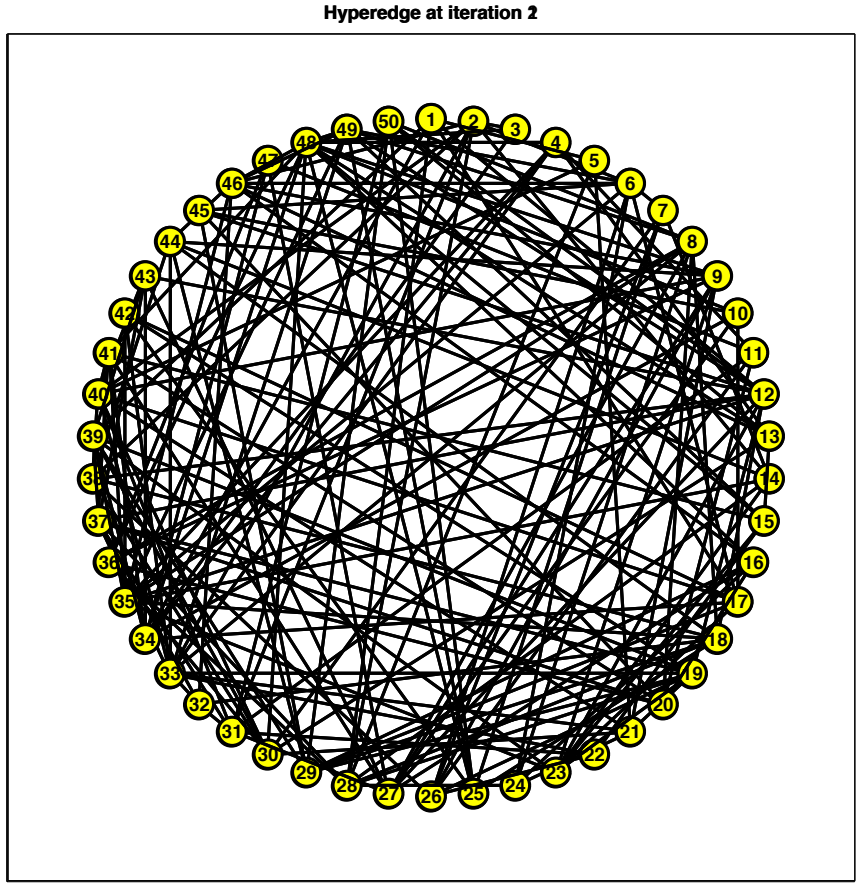
$d = 2$ items/sale

$s = 250$ sale patterns

- total cut values $2^n = 2^{50}$
- sparsity $K \leq s2^{d-1} = 500$
- # of cut queries $O(Kn) \approx 25000$

# WHT – Hypergraph Sketching





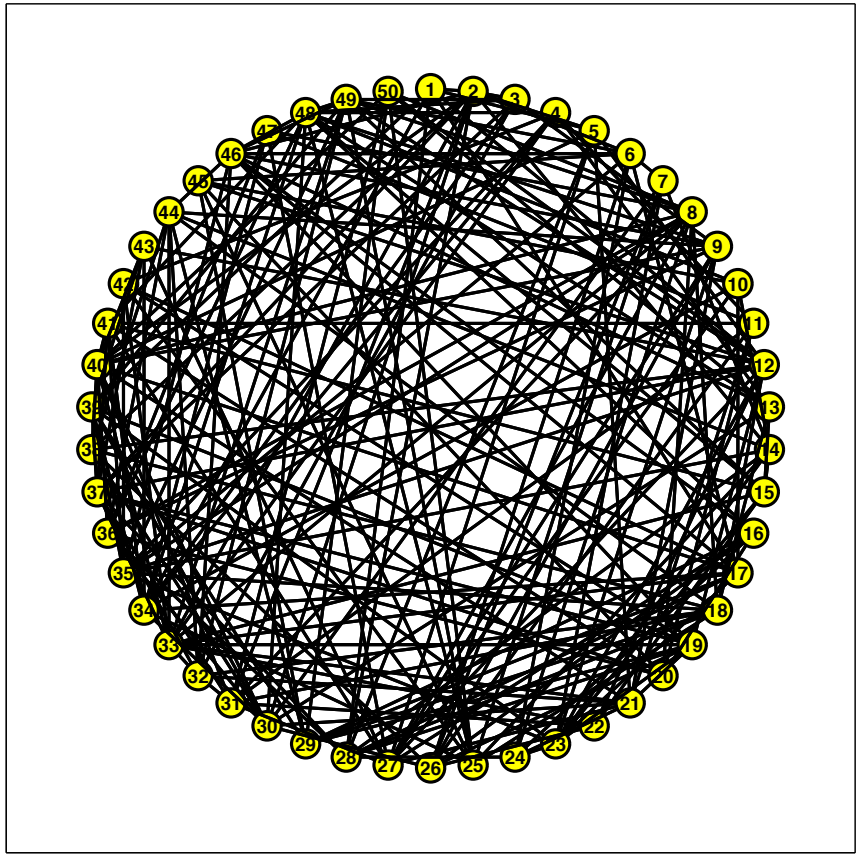

$n = 50$ books

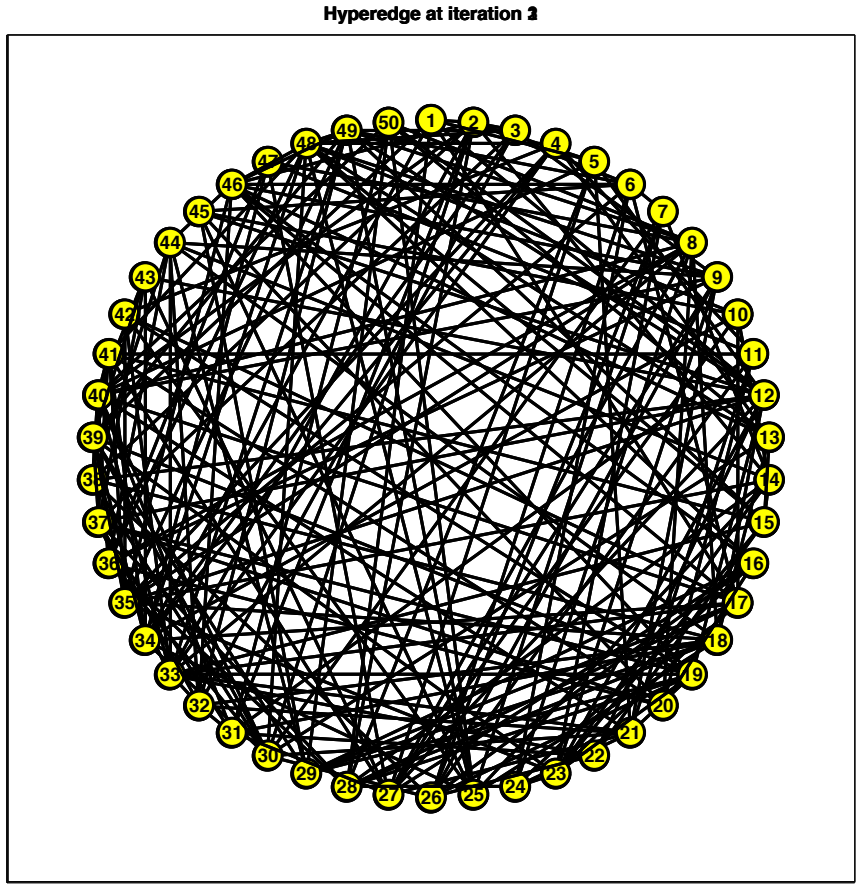
$d = 2$ items/sale

$s = 250$ sale patterns

- total cut values $2^n = 2^{50}$
- sparsity $K \leq s2^{d-1} = 500$
- # of cut queries $O(Kn) \approx 25000$

# WHT – Hypergraph Sketching





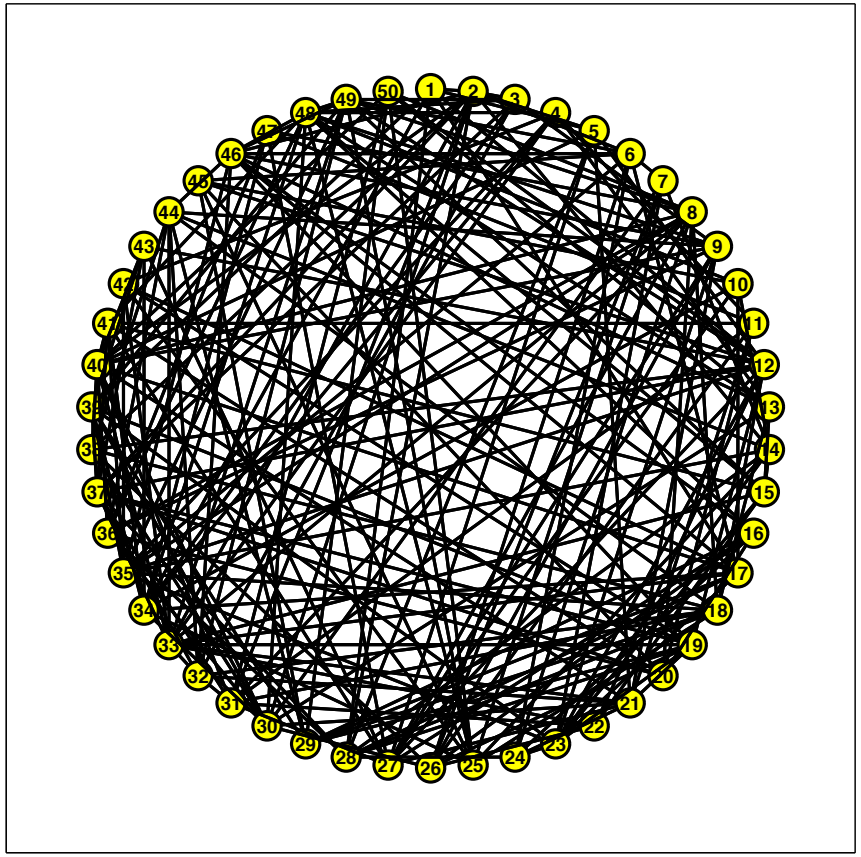

$n = 50$ books

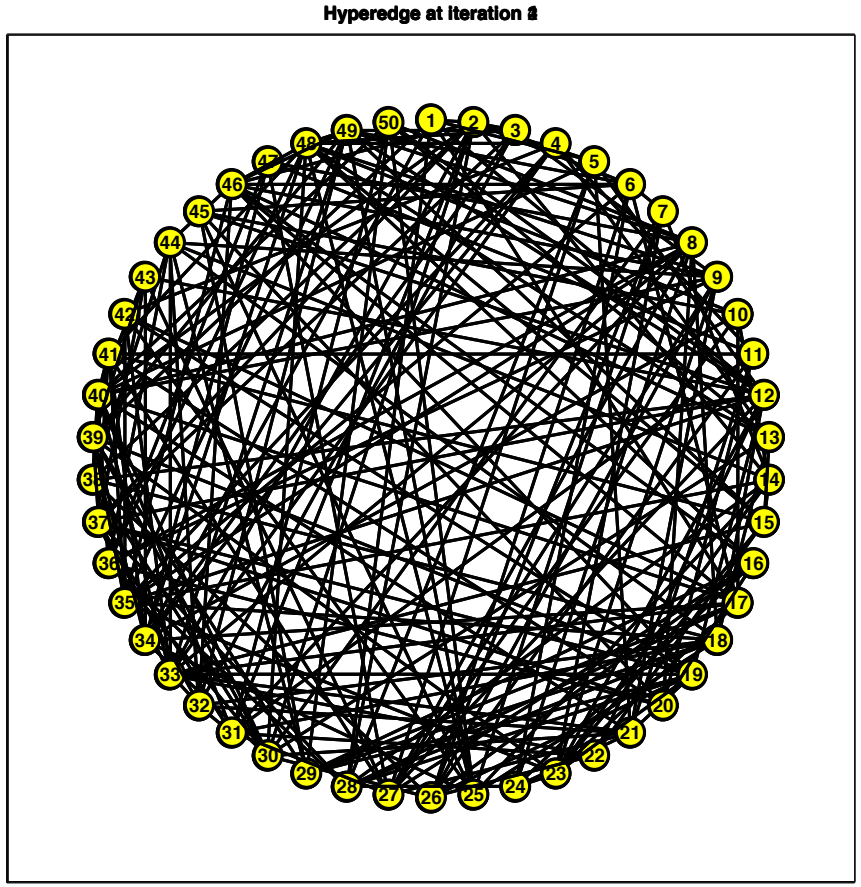
$d = 2$ items/sale

$s = 250$ sale patterns

- total cut values $2^n = 2^{50}$
- sparsity $K \leq s2^{d-1} = 500$
- # of cut queries $O(Kn) \approx 25000$

# WHT – Hypergraph Sketching






$n = 50$ books

$d = 2$ items/sale

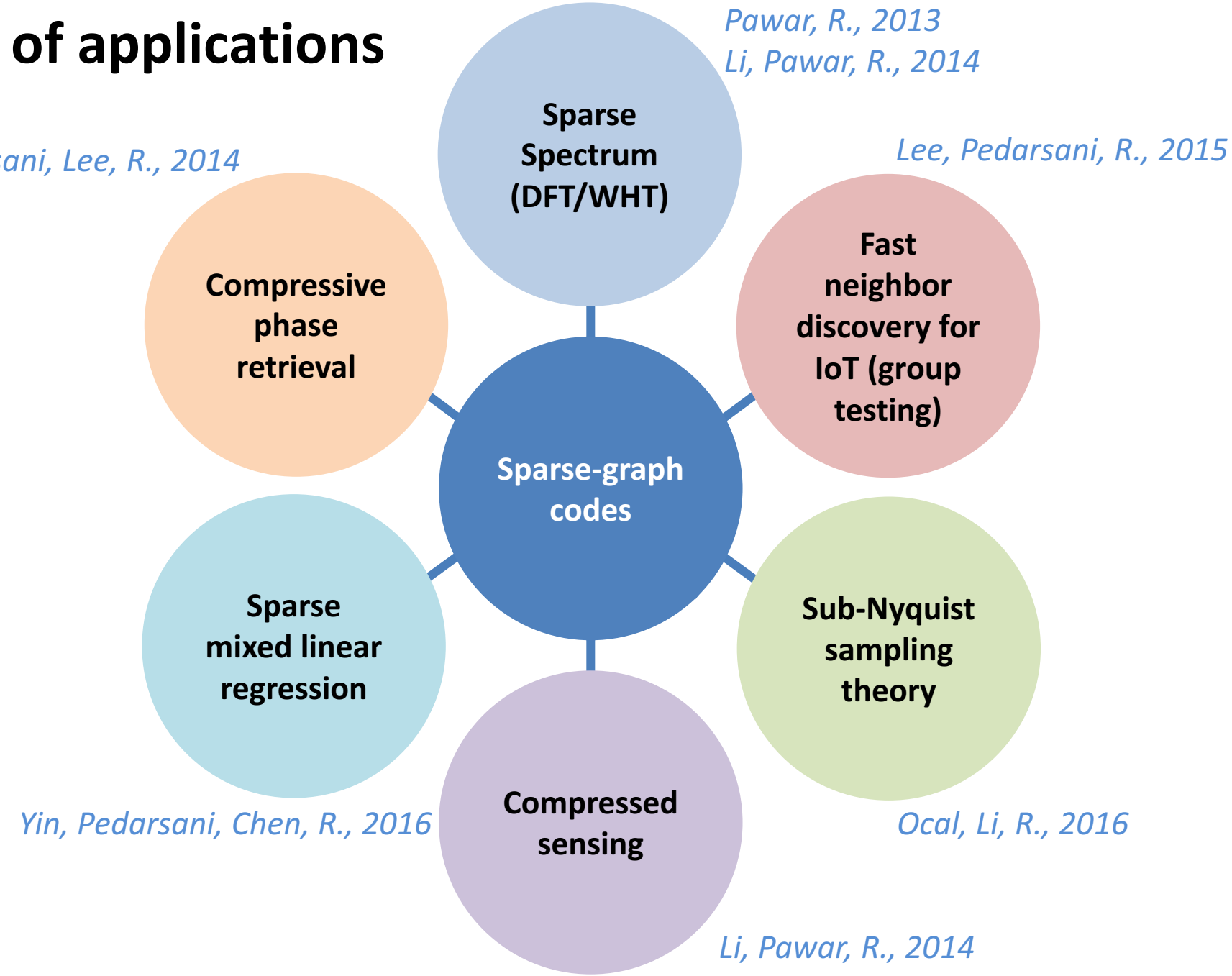
$s = 250$ sale patterns

- total cut values $2^n = 2^{50}$
- sparsity $K \leq s2^{d-1} = 500$
- # of cut queries $O(Kn) \approx 25000$

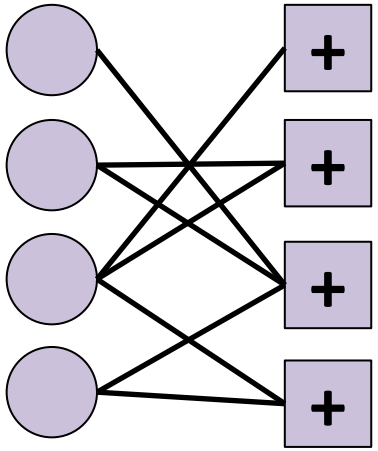
# Open source implementations

- **Sparse FFT and Sparse WHT** implemented in C++
- Publicly available on GitHub https://github.com/ucbasics

# Broad scope of applications



# Unifying separation architecture for sparse-signal processing




Sparse-Graph Code

"Peeling-based" turbo engine

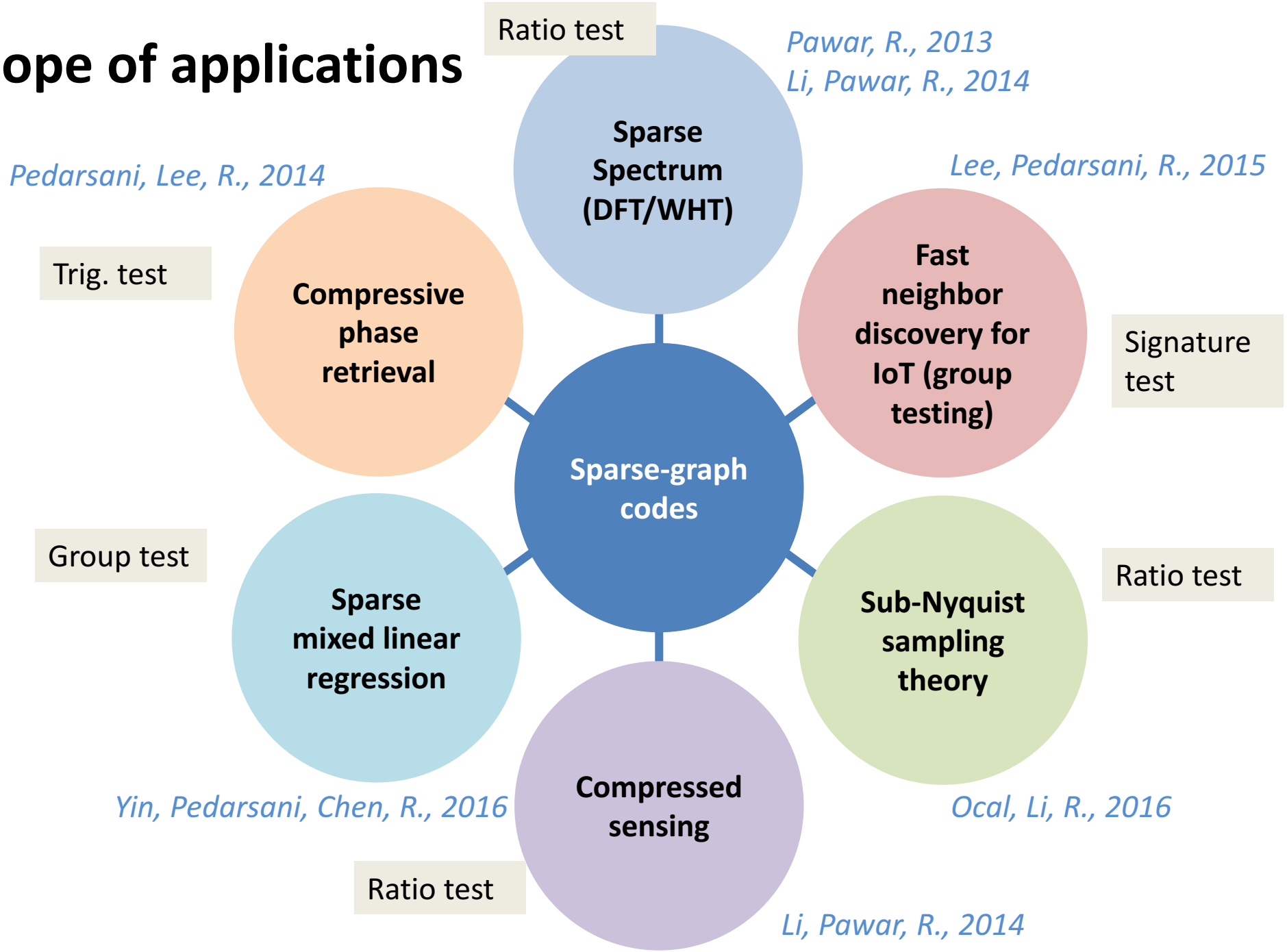
Divide

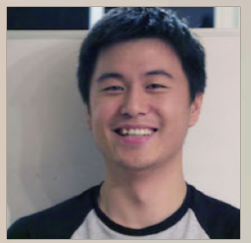
Concur

"Solve-if-trivial" sub-engine

Problem-specific sub-code

# Broad scope of applications

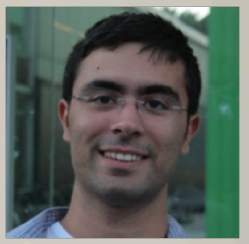




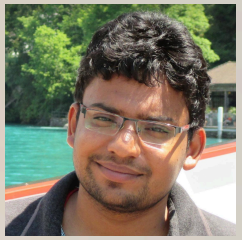
Kangwook Lee

Ramtin P.

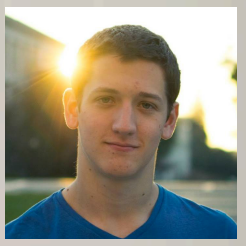
Dimitris P.



Orhan Ocal



Vipul Gupta

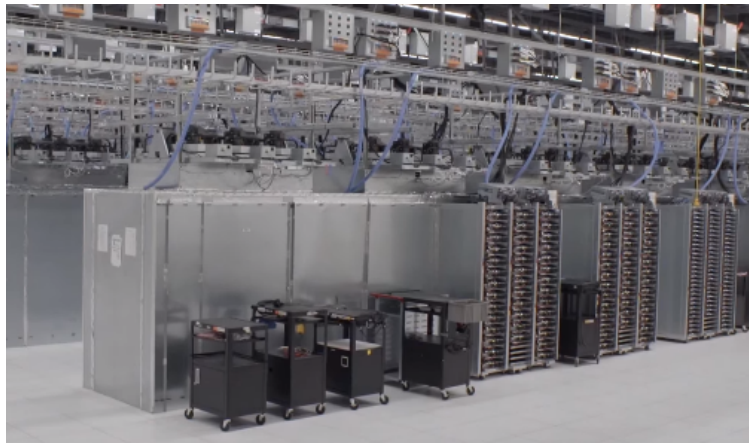


Tavor Baharav

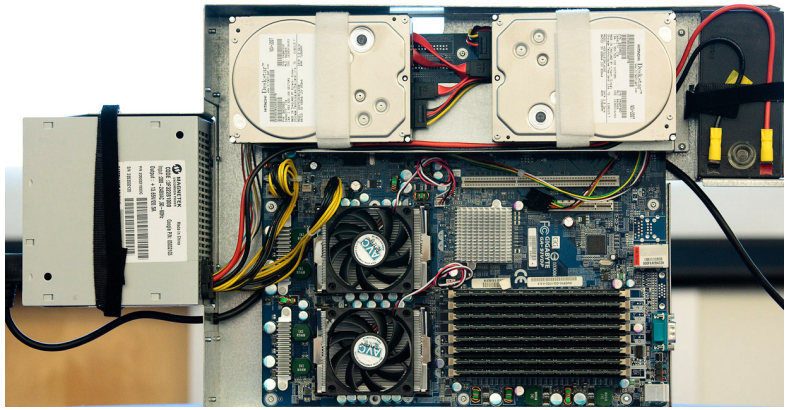
# Chapter 3

## Speeding up distributed computing on the cloud

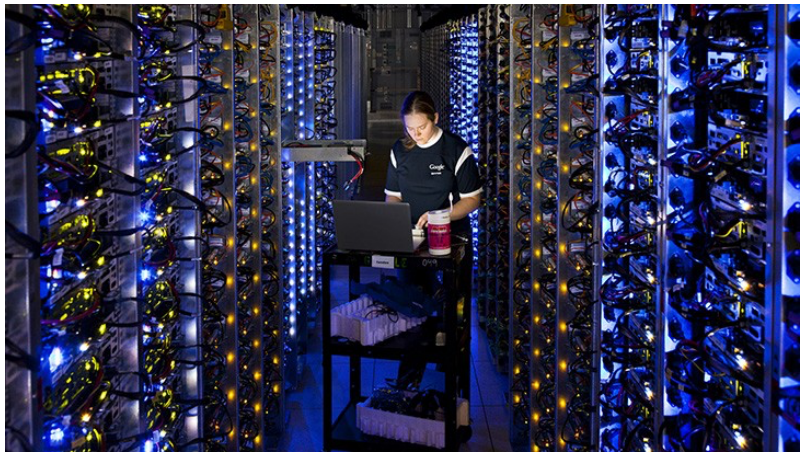
# System Noise



Network bottlenecks

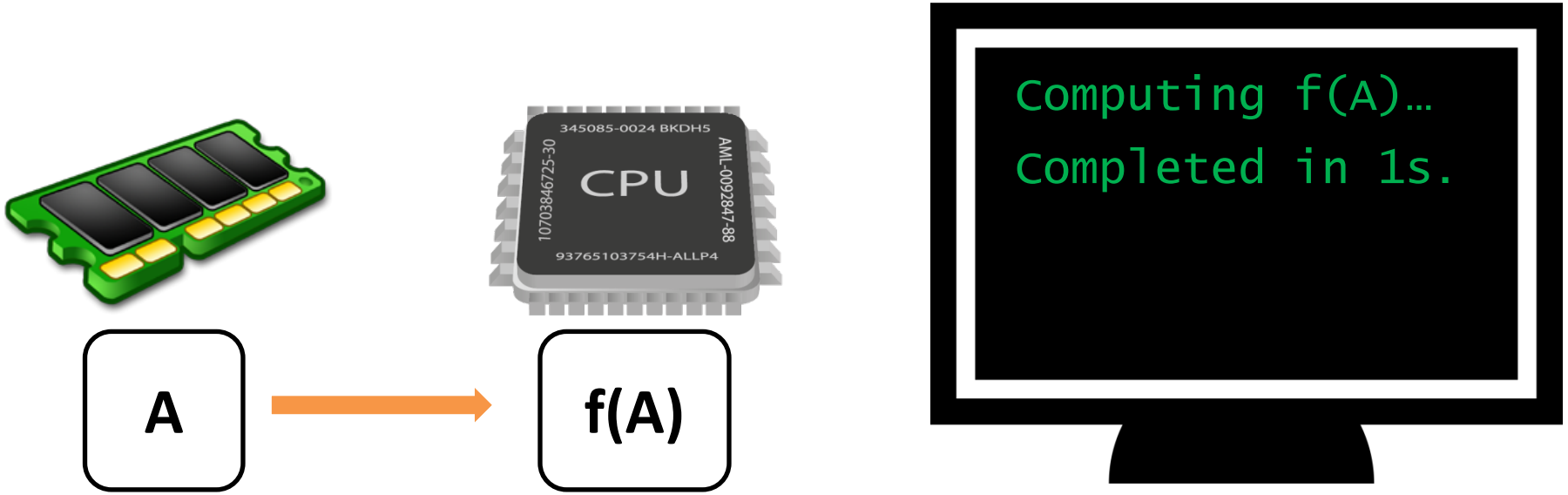


HW failures

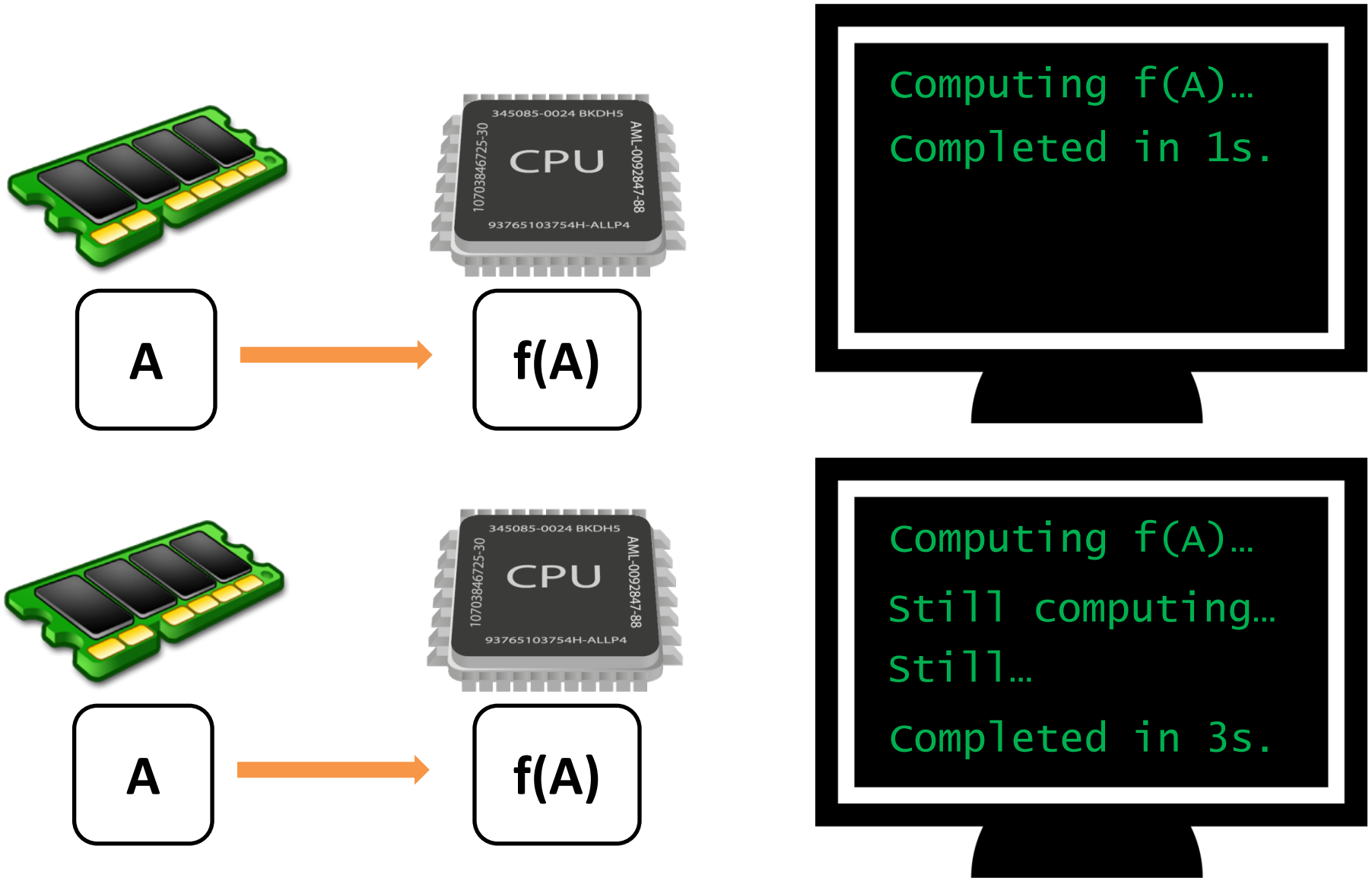


Maintenance, etc.

# System Noise = Latency Variability



# System Noise = Latency Variability



# Straggler Problem: Data Centers

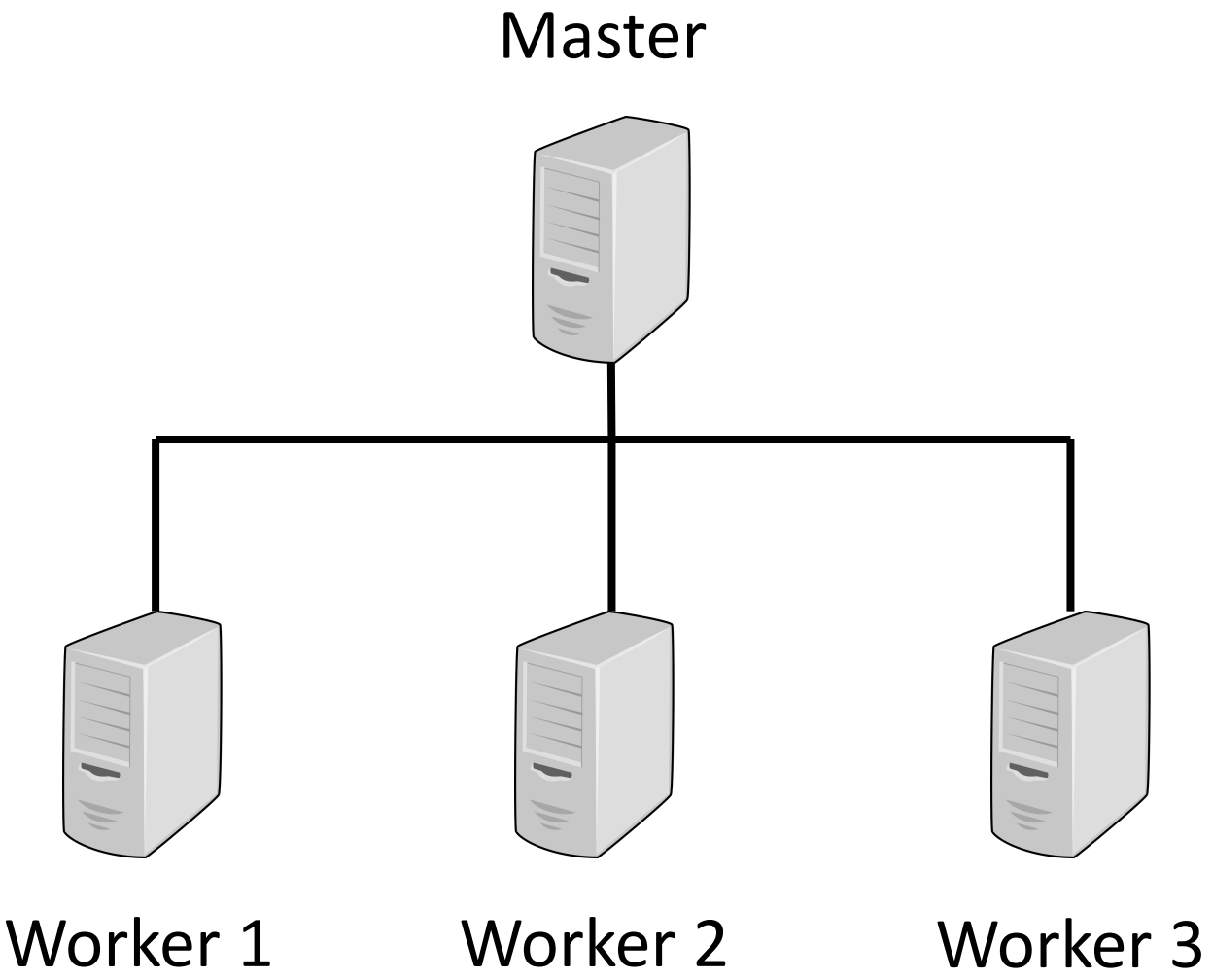
- "The Tail at Scale", Comm. of the ACM 2013

***"The scale and complexity of modern Web services make it infeasible to eliminate all latency variability."***

| | 50%ile latency | 95%ile latency | 99%ile latency |
|---|---|---|---|
| One random leaf finishes | 1ms | 5ms | 10ms |
| 95% of all leaf requests finish | 12ms | 32ms | 70ms |
| 100% of all leaf requests finish | 40ms | 87ms | 140ms |

# Distributed Matrix-Vector Multiplication

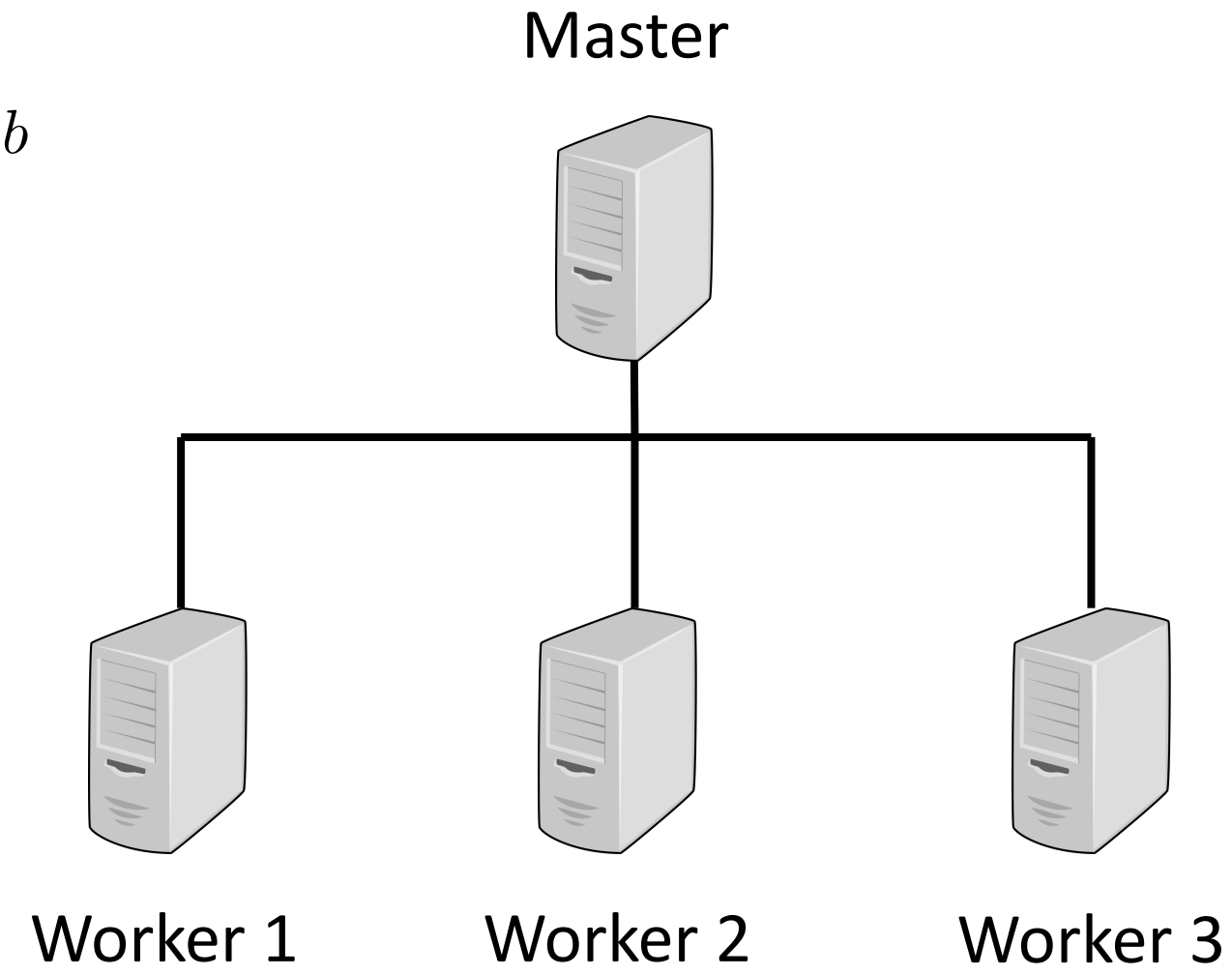
$A \times b$



# Distributed Matrix-Vector Multiplication

$$A \times b$$

$$= \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \times b$$

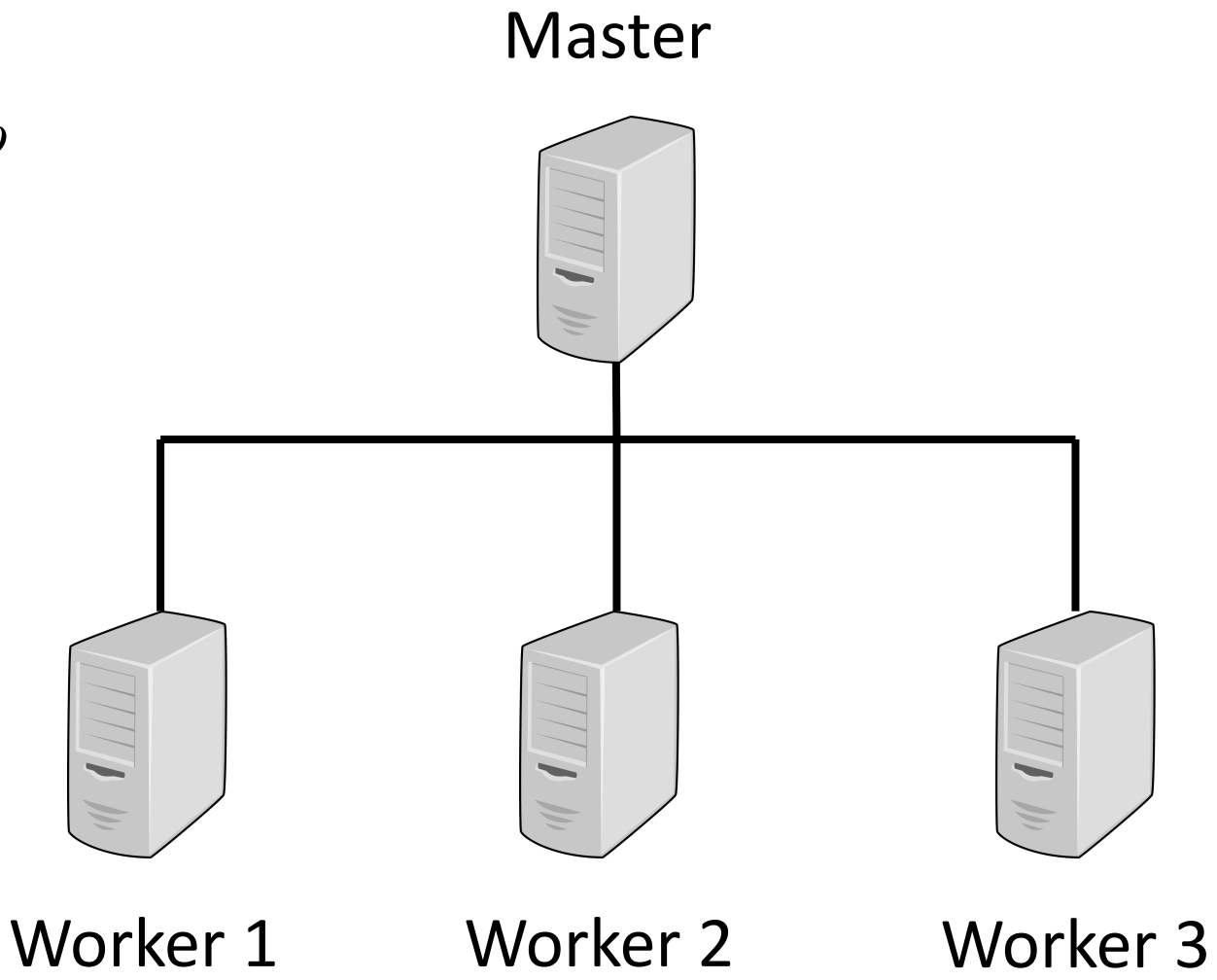


# Distributed Matrix-Vector Multiplication

$$A \times b$$

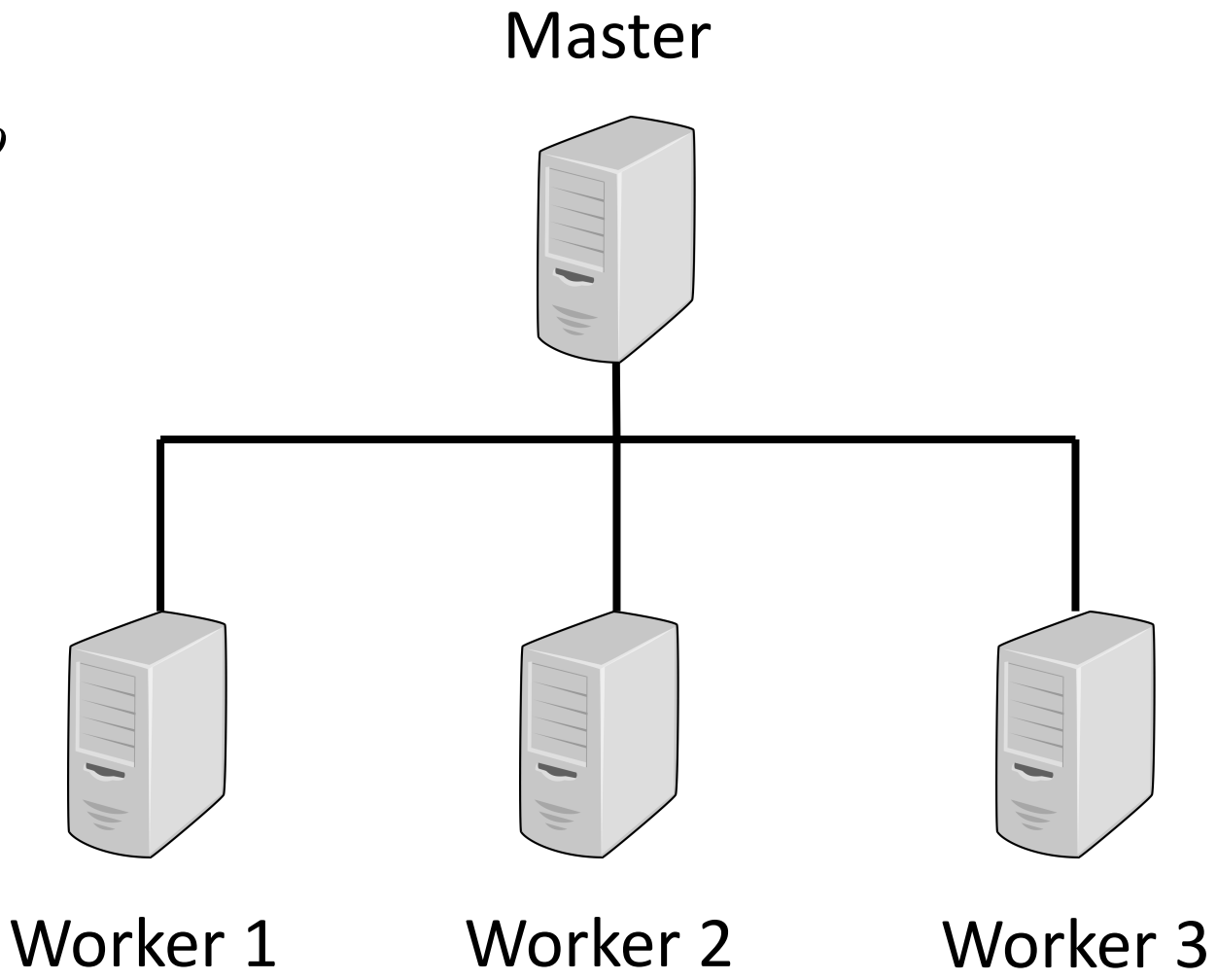
$$= \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \times b$$

$$= \begin{pmatrix} A_1 \times b \\ A_2 \times b \\ A_3 \times b \end{pmatrix}$$



# Distributed Matrix-Vector Multiplication

$$
\begin{aligned}
& A \times b \\
&= \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \times b \\
&= \begin{pmatrix} A_1 \times b \\ A_2 \times b \\ A_3 \times b \end{pmatrix} \\
&:= \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}
\end{aligned}
$$



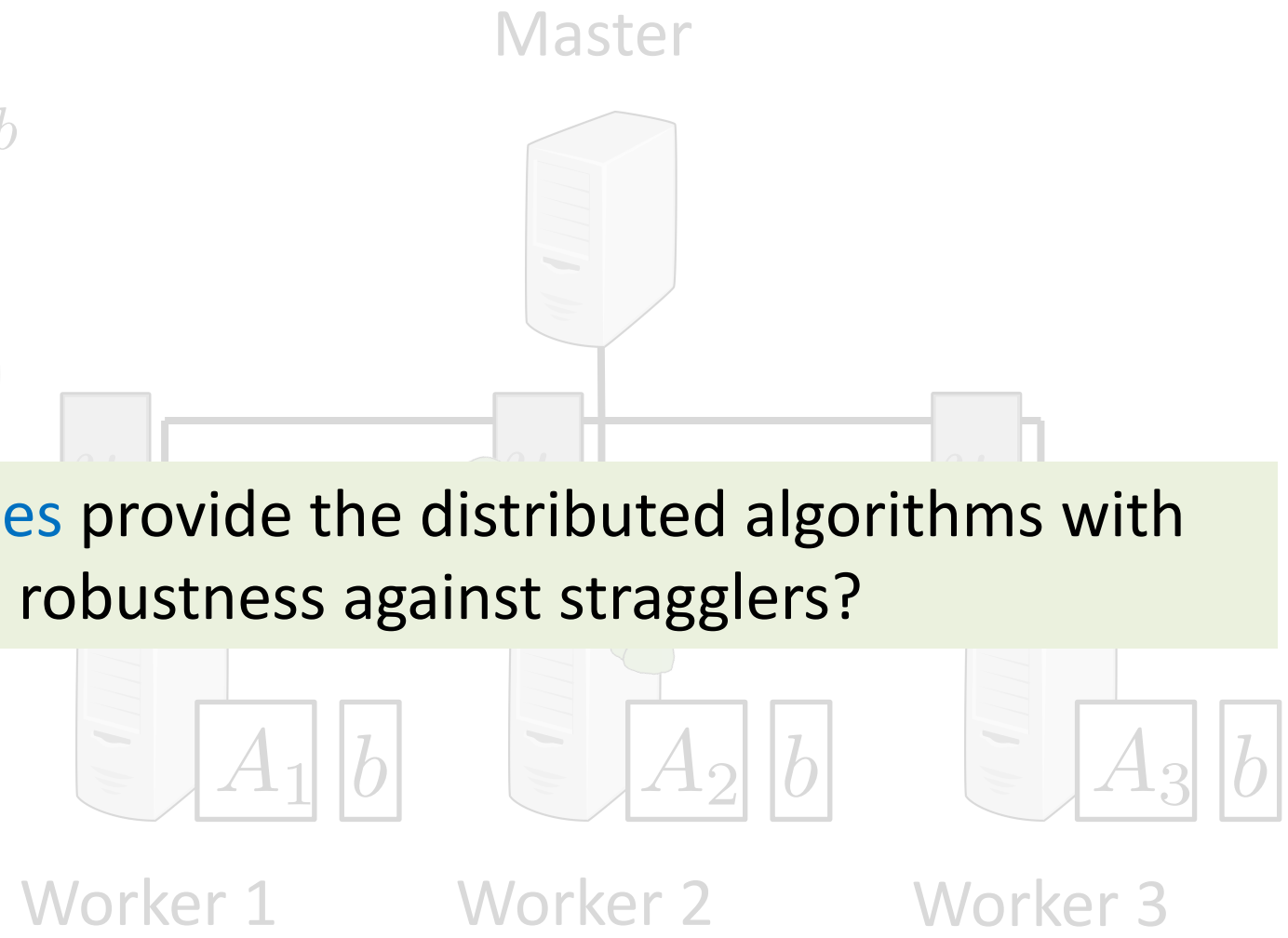
# Distributed Matrix-Vector Multiplication

$$A \times b$$

$$= \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \times b$$

$$= \begin{pmatrix} A_1 \times b \\ A_2 \times b \\ A_3 \times b \end{pmatrix}$$

$$\begin{pmatrix} y_3 \end{pmatrix}$$

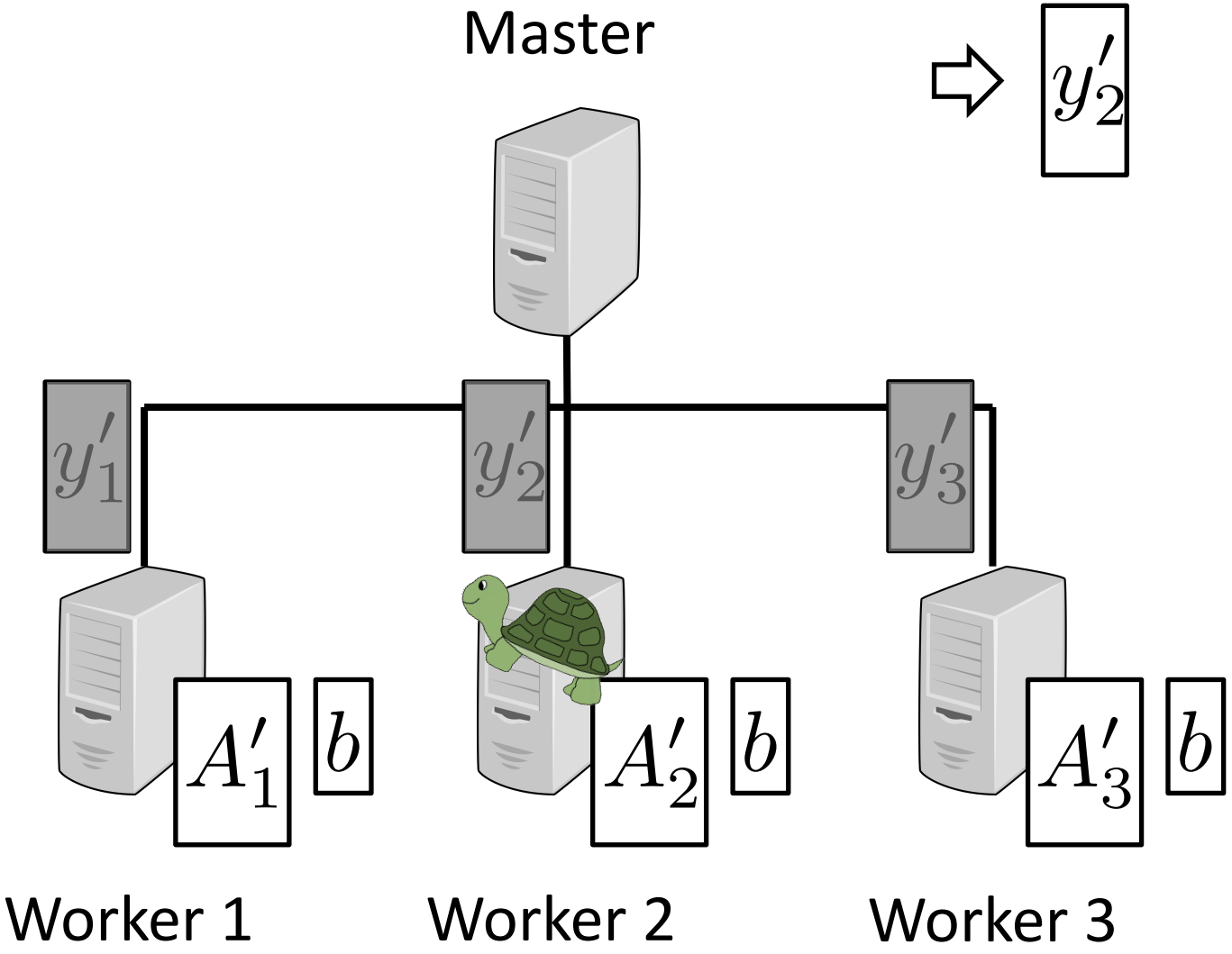



**Q.** Can codes provide the distributed algorithms with robustness against stragglers?

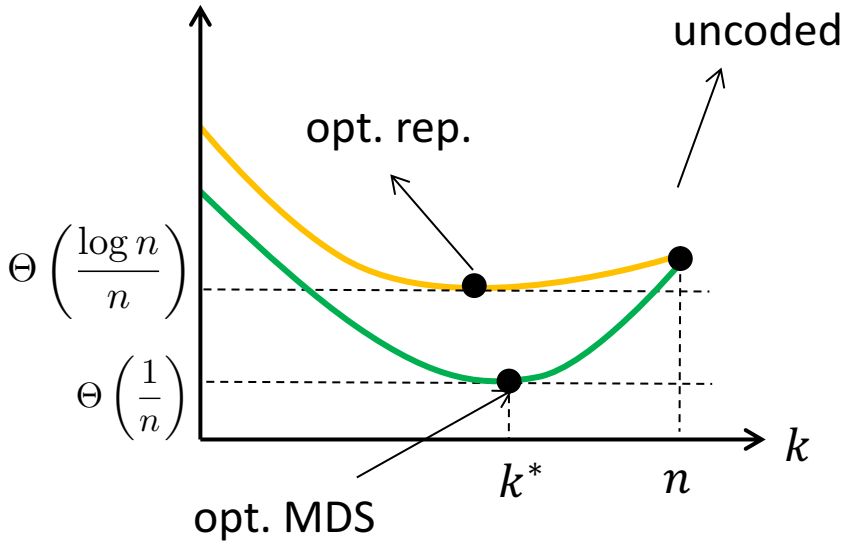
# Coded Matrix-Vector Multiplication

$$A \times b$$
$$= \begin{pmatrix} A'_1 \\ A'_2 \end{pmatrix} \times b$$
$$= \begin{pmatrix} A'_1 \times b \\ A'_2 \times b \end{pmatrix}$$
$$:= \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix}$$

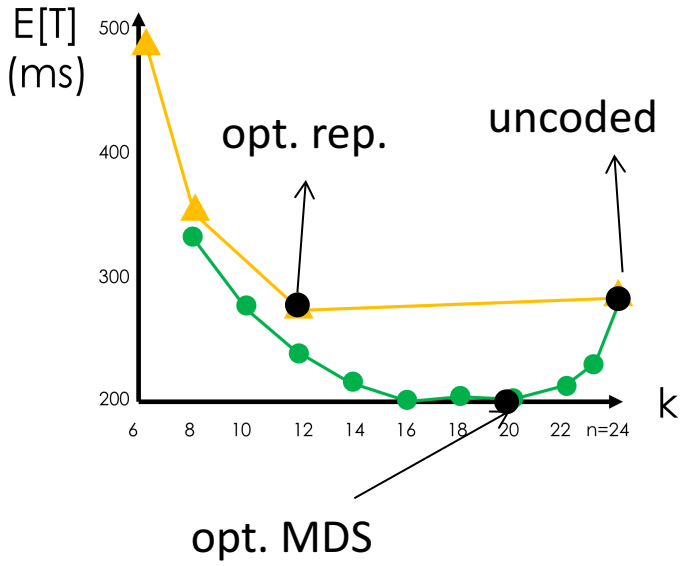
$$A'_3 := A'_1 + A'_2$$
$$y'_3 := y'_1 + y'_2$$



# MDS-Coded Matrix-Vector Multiplication




Under exponential latency model




On Amazon AWS

[LLPP**R**, NIPS workshop'15]
[LLPP**R**, T-IT'18]

# Applications

- Distributed linear or logistic regression
- Distributed non-linear function computation
- Reducing communication in data shuffling by network coding

*Has attracted lots of interest:*

- ***Coded Matrix Multiplication***
- ***Distributed Gradient Coding***
- **Approximation:**

  ***SVD + Sequential Coding, Sketching, Second-order methods, Coded Control***
- ***…***

# Computing on the Cloud with PyWren (AWS Lambda)

*Jonas et al., "Occupy the Cloud, distributed computing for the 99%" (2017)*

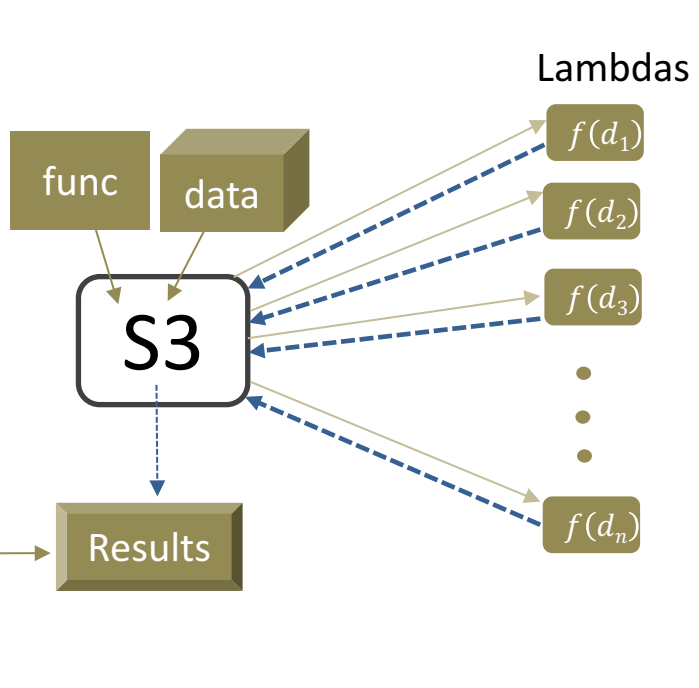
Recent interest in "microservices" has led to Amazon's AWS Lambda, where each lambda task gets:
- a single core (exact performance varies a bit)
- 1536 MB of RAM max and 300s max execution time & a runtime of python or Java

Your laptop

The cloud

Write a function
Upload the function and data to S3
Invoke Lambda machines

Poll S3 for results



# AWS Lambda


**PyWren**: Can have up to 16,000 workers on AWS Lambda

# Scalable Coded Computing

- **Latency = Enc. Time** + **Computation Time** + **Dec. Time** (plus comm. cost to move data in distributed setting)
- MDS codes for matrix-vector (Lee et al. '15) or matrix-matrix (poly. codes: Yu et al. '17) target only the **computation time**
- At scale? (e.g., Lambda w/6K workers & 400+ stragglers)
- **Decoding "on the fly":** real-valued computations

**Product Codes:** a good tradeoff bet. near-MDS and local enc./dec.

# Conclusions

- Codes for **speed**: offer a **CLEAR** advantage – **C**omputation, **LE**arning, **A**ccess, **R**ecovery.

- **A**ccess: Regeneration codes and beyond

- **LE**arning & **R**ecovery: Sparse-graph codes based "peeling" core with "verification" test

- **C**omputation: Straggler-proofing large-scale distributed computing


Thank You!
QUESTIONS?