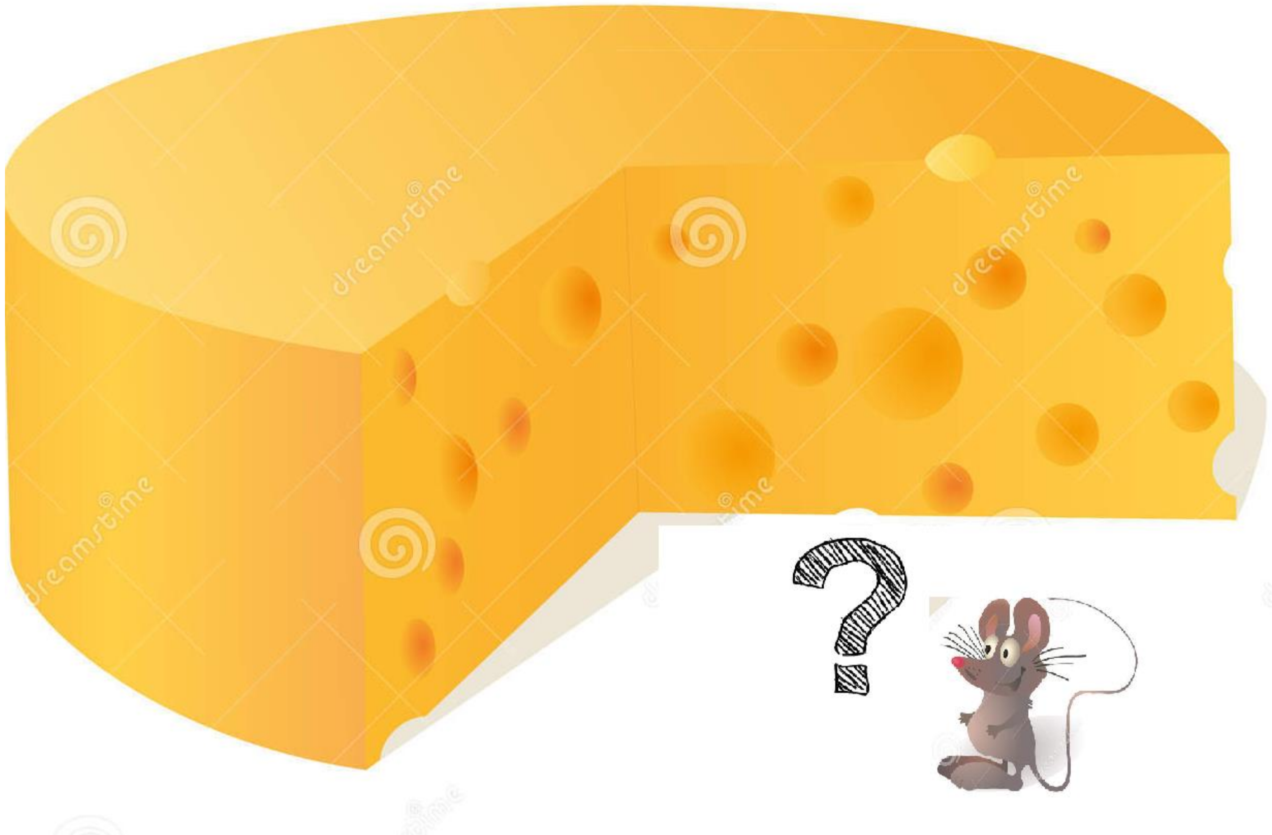
Big Data Class



LECTURER: DAN FELDMAN

TEACHING ASSISTANTS:

IBRAHIM JUBRAN

ALAA MAALOUF

אוניברסיטת חיפה
University of Haifa
جامعة حيفا

Traditional Measurements

Input size: n

Running time: $t(n)$

Memory (space): $s(n)$

In CS and in this class we use the "O“ notation:
- We do not try to optimize constants.

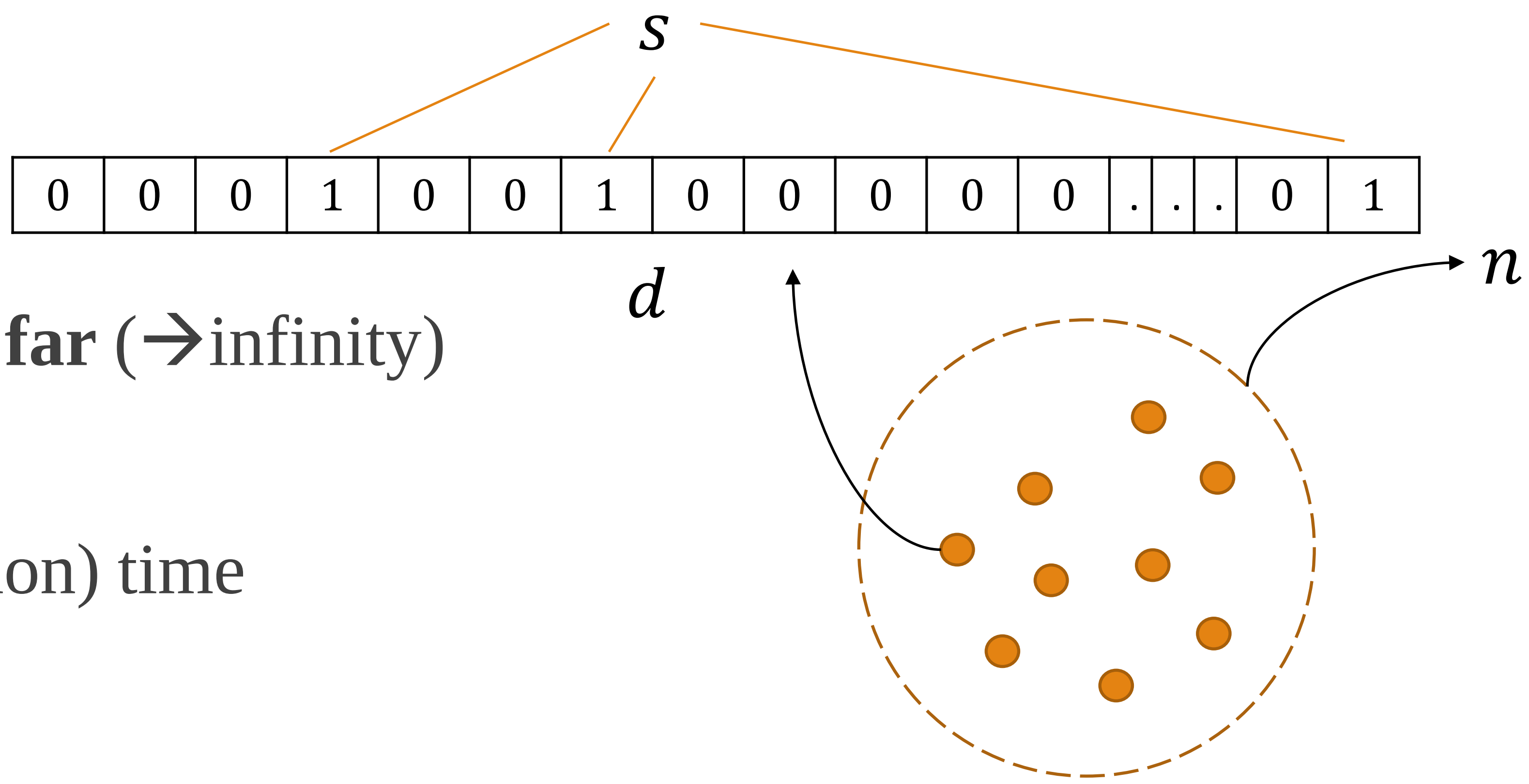
$n \to \infty$ (justifies not handling constants).

“Memory”: usually refers to RAM. Might also be HDD.
- Memory for running an algorithm might be much larger than n (the input size).

Big data Input

Data dimensionality (# of features): d

Data sparsity: s



n = # of points seen **so far** (→infinity)

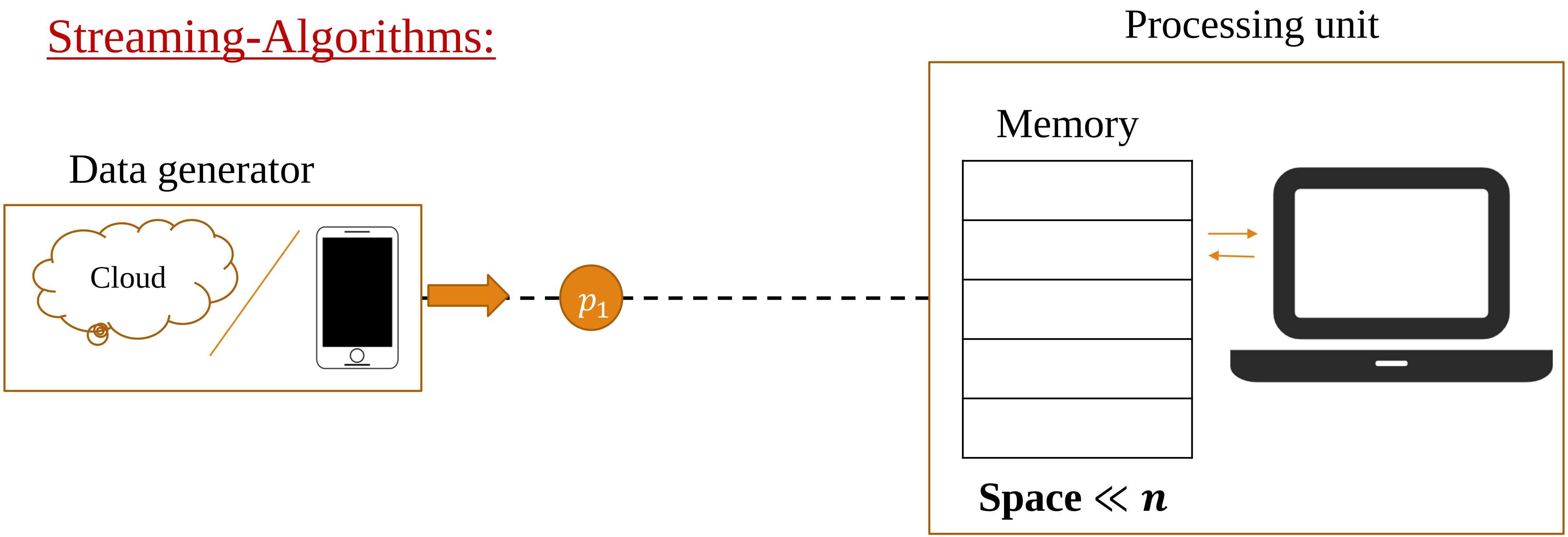
M = # of machines

update (insertion/deletion) time

Per coordinate/item

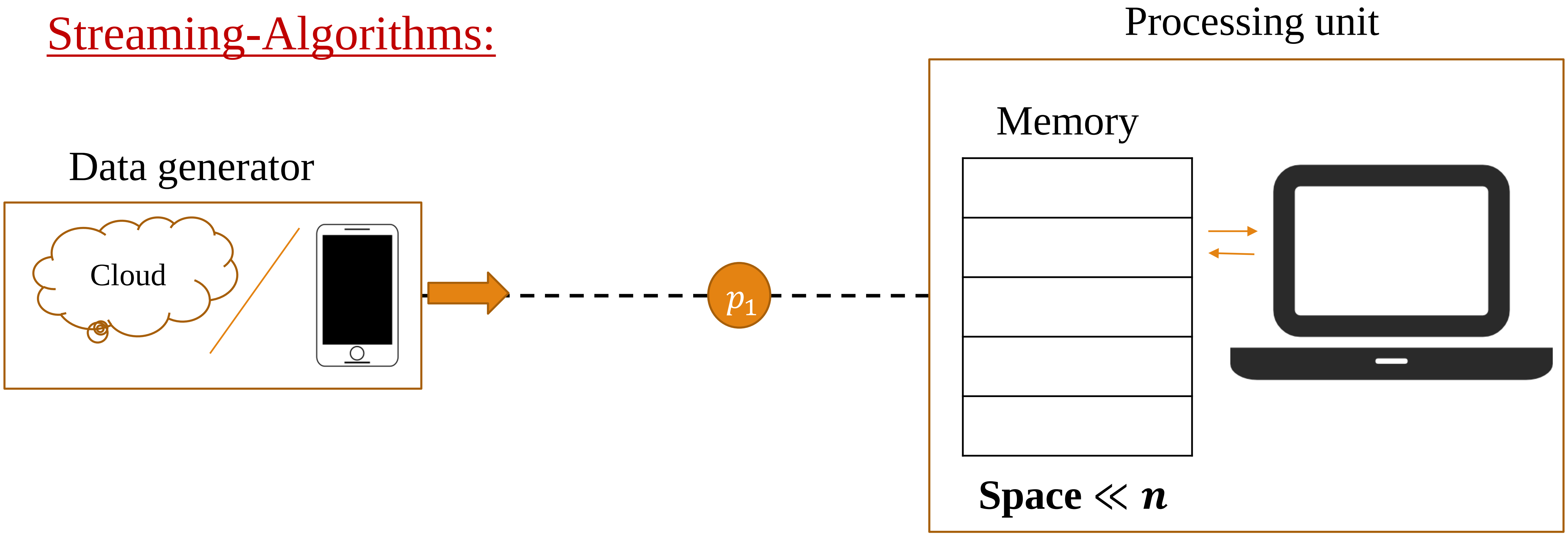
Big data models

Streaming-Algorithms:



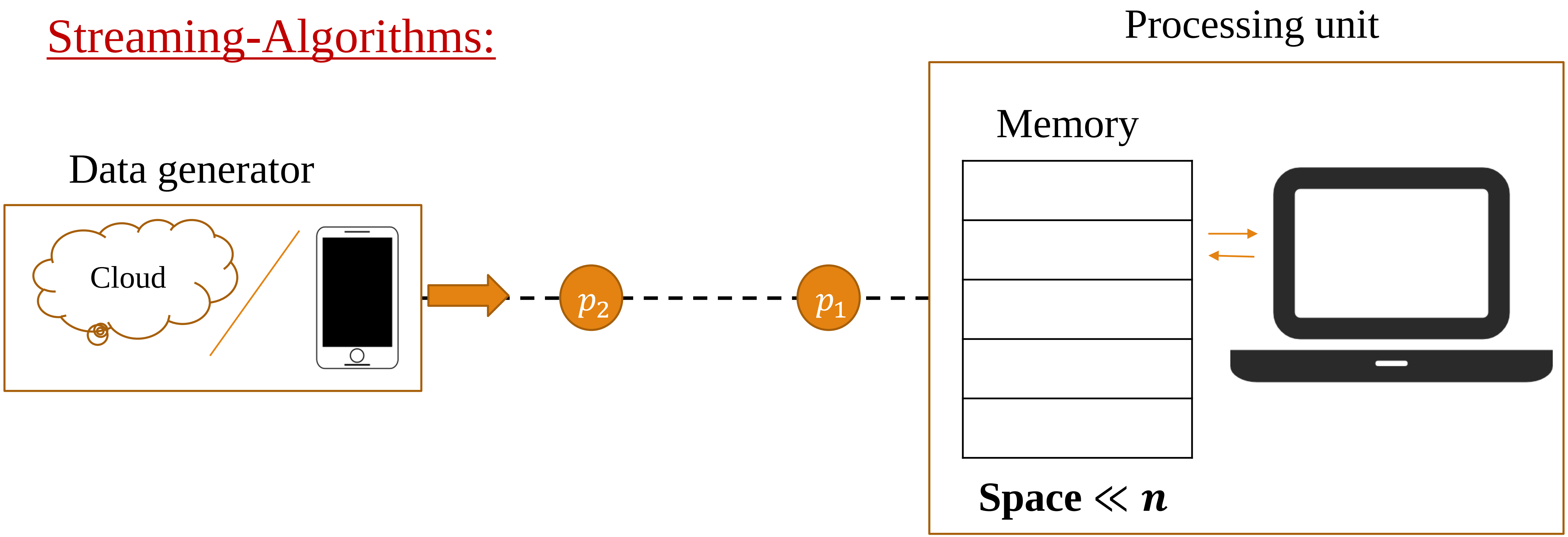
Big data models

Streaming-Algorithms:

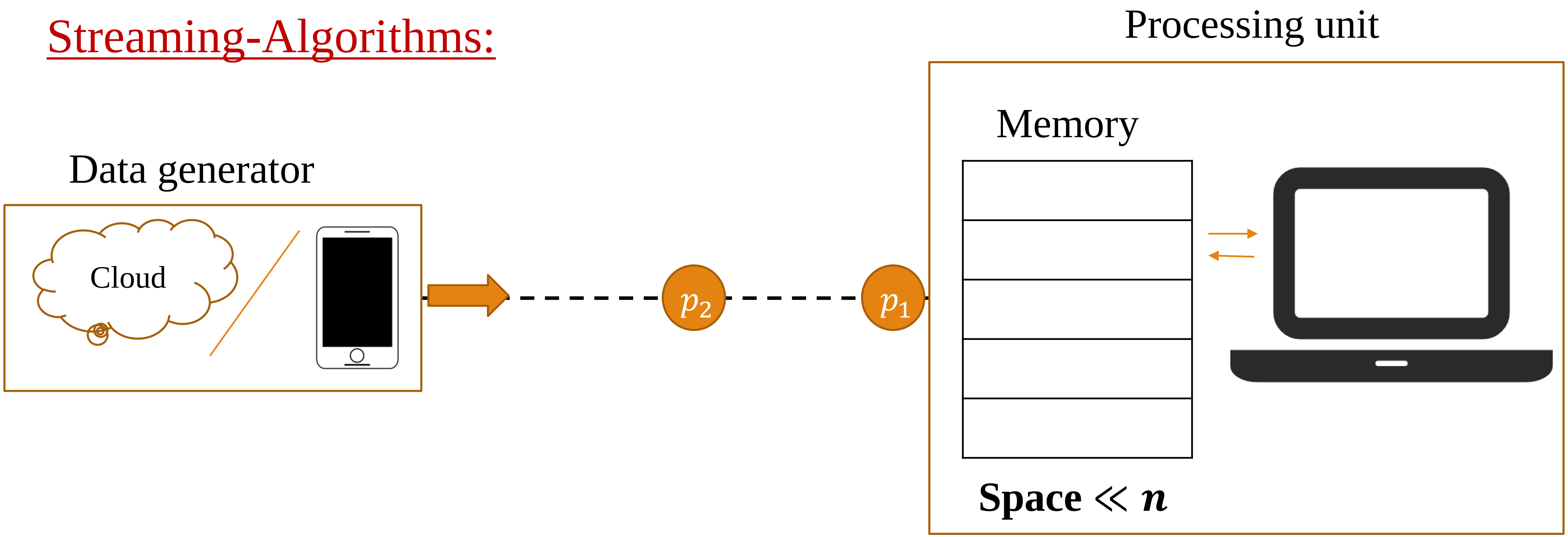


Big data models

Streaming-Algorithms:

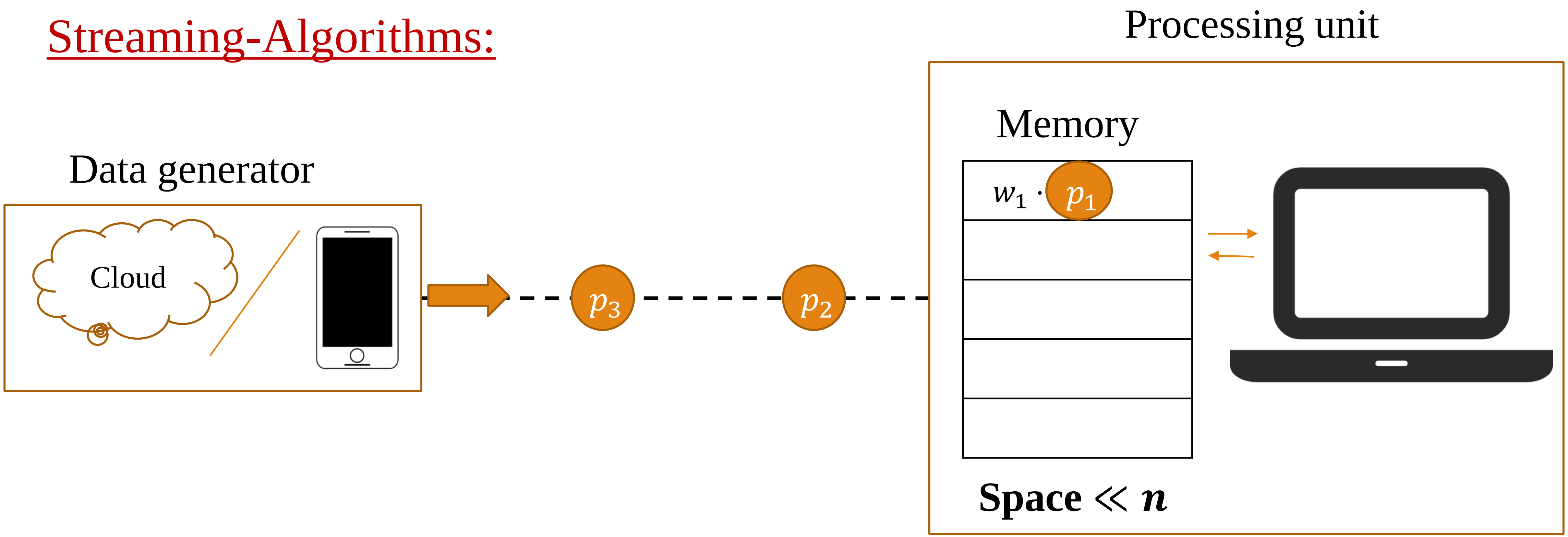


Big data models



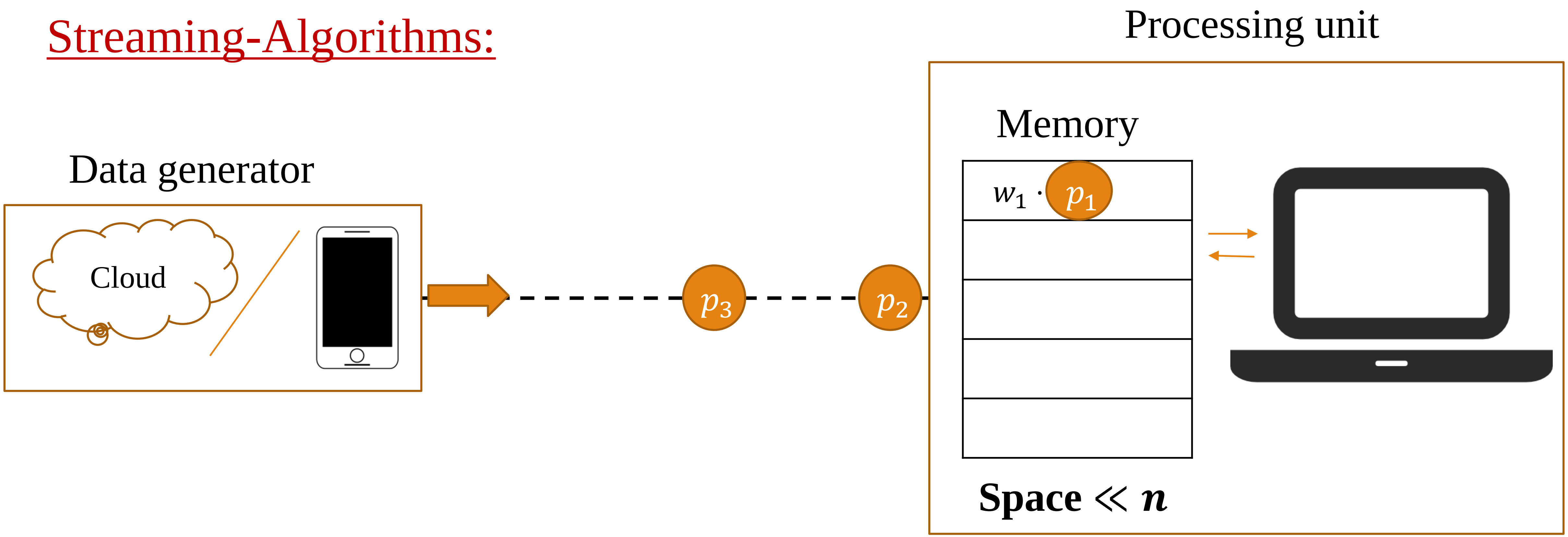
Big data models

Streaming-Algorithms:



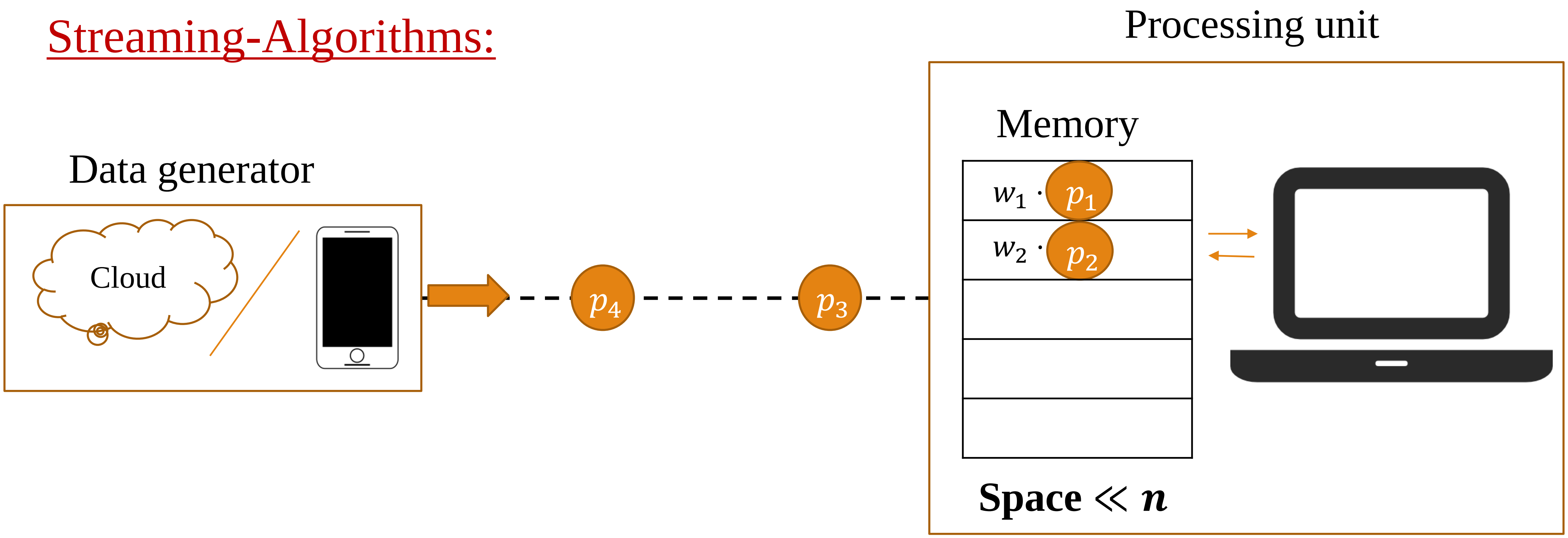
Big data models

Streaming-Algorithms:



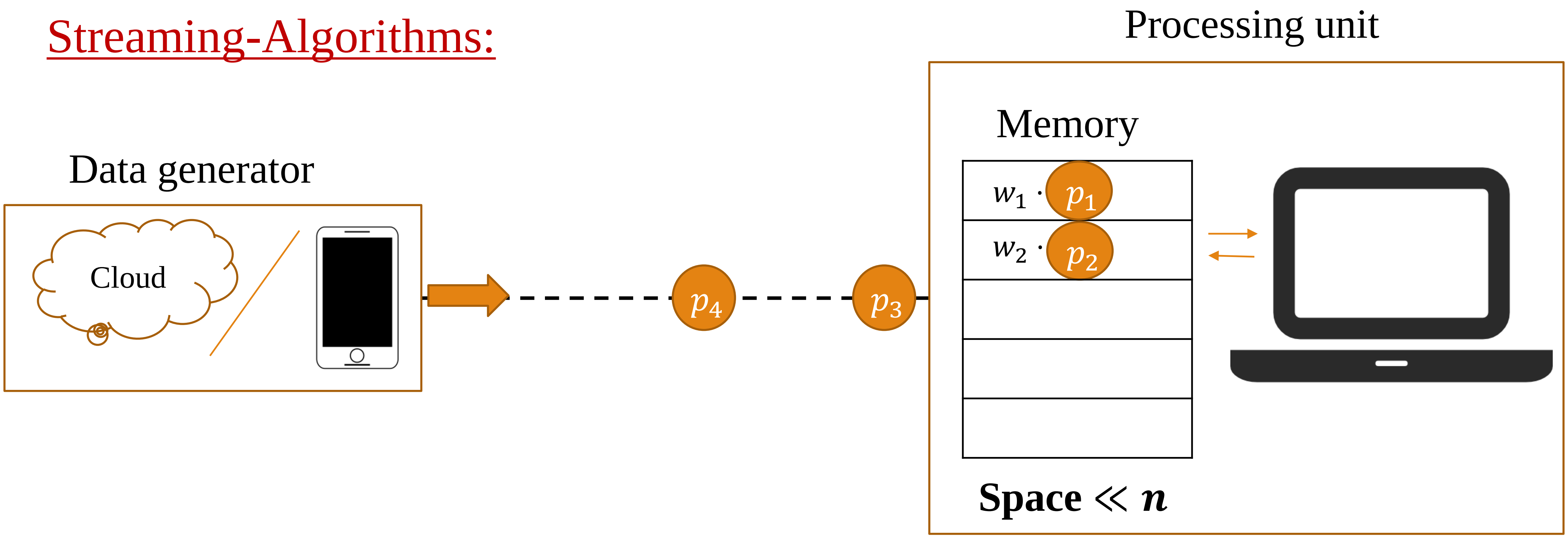
Big data models

Streaming-Algorithms:



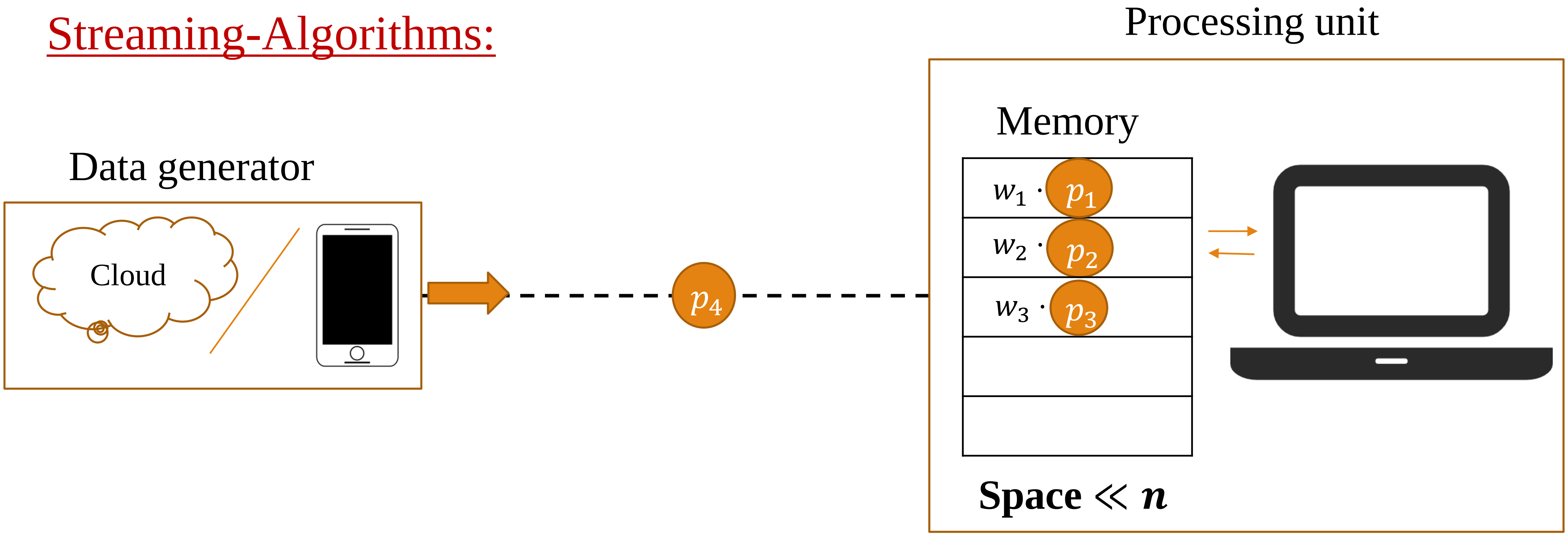
Big data models

Streaming-Algorithms:



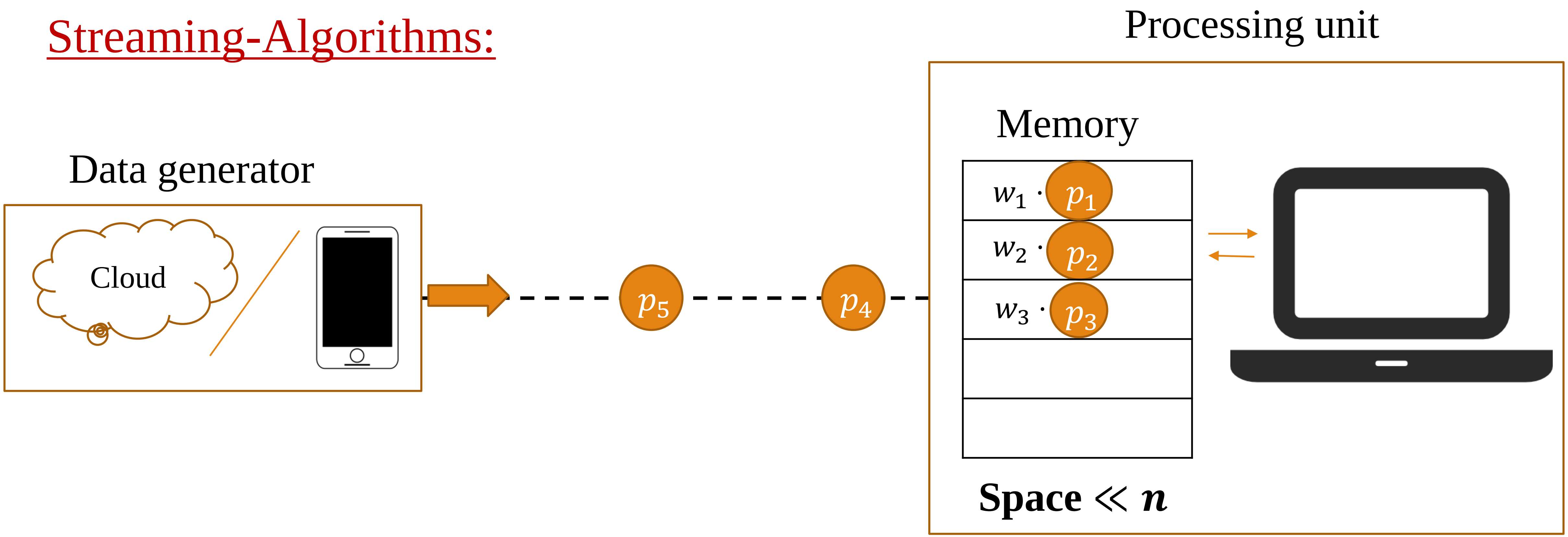
Big data models

Streaming-Algorithms:



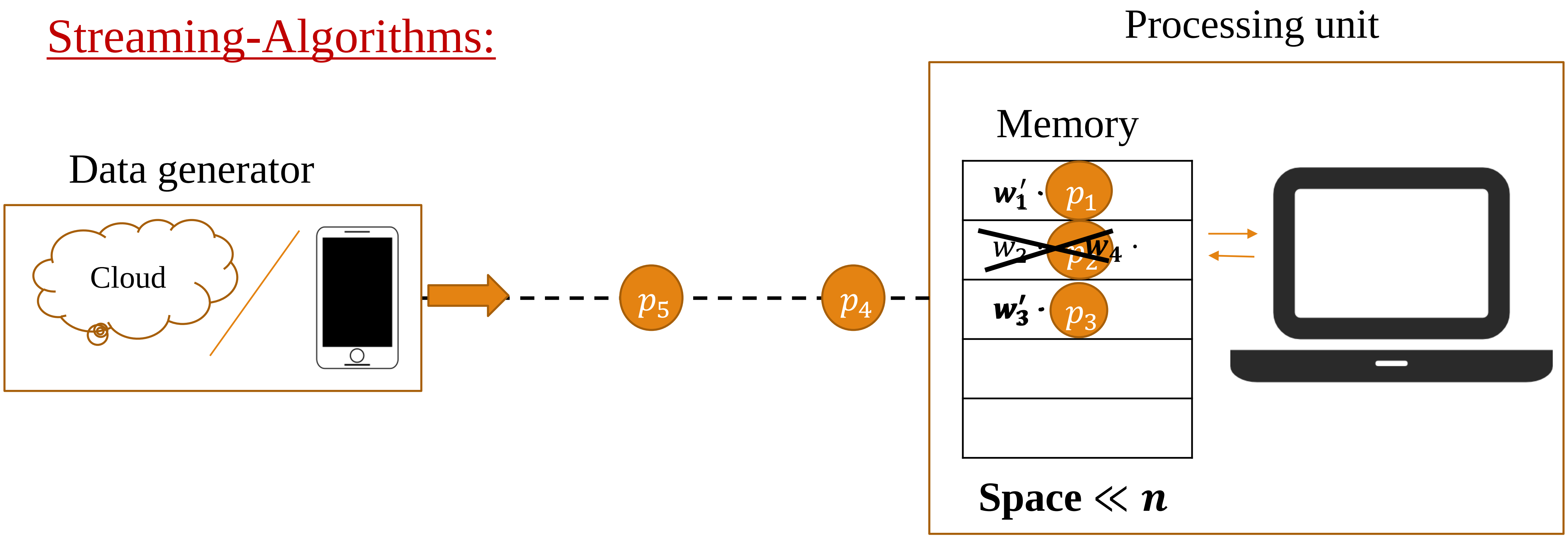
Big data models

<u>Streaming-Algorithms:</u>



Big data models

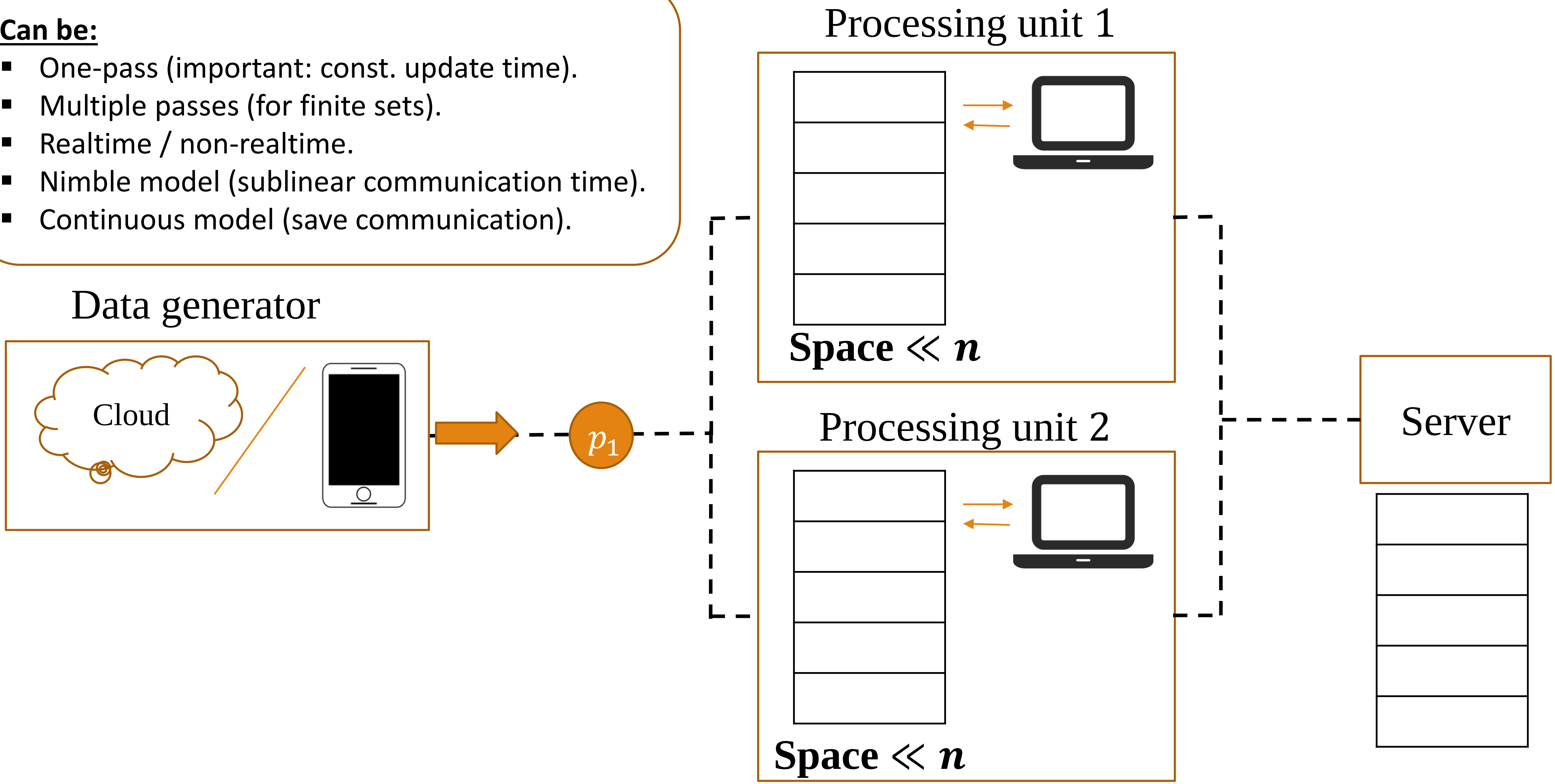
Streaming-Algorithms:



Streaming and distributed algorithms:

Can be:

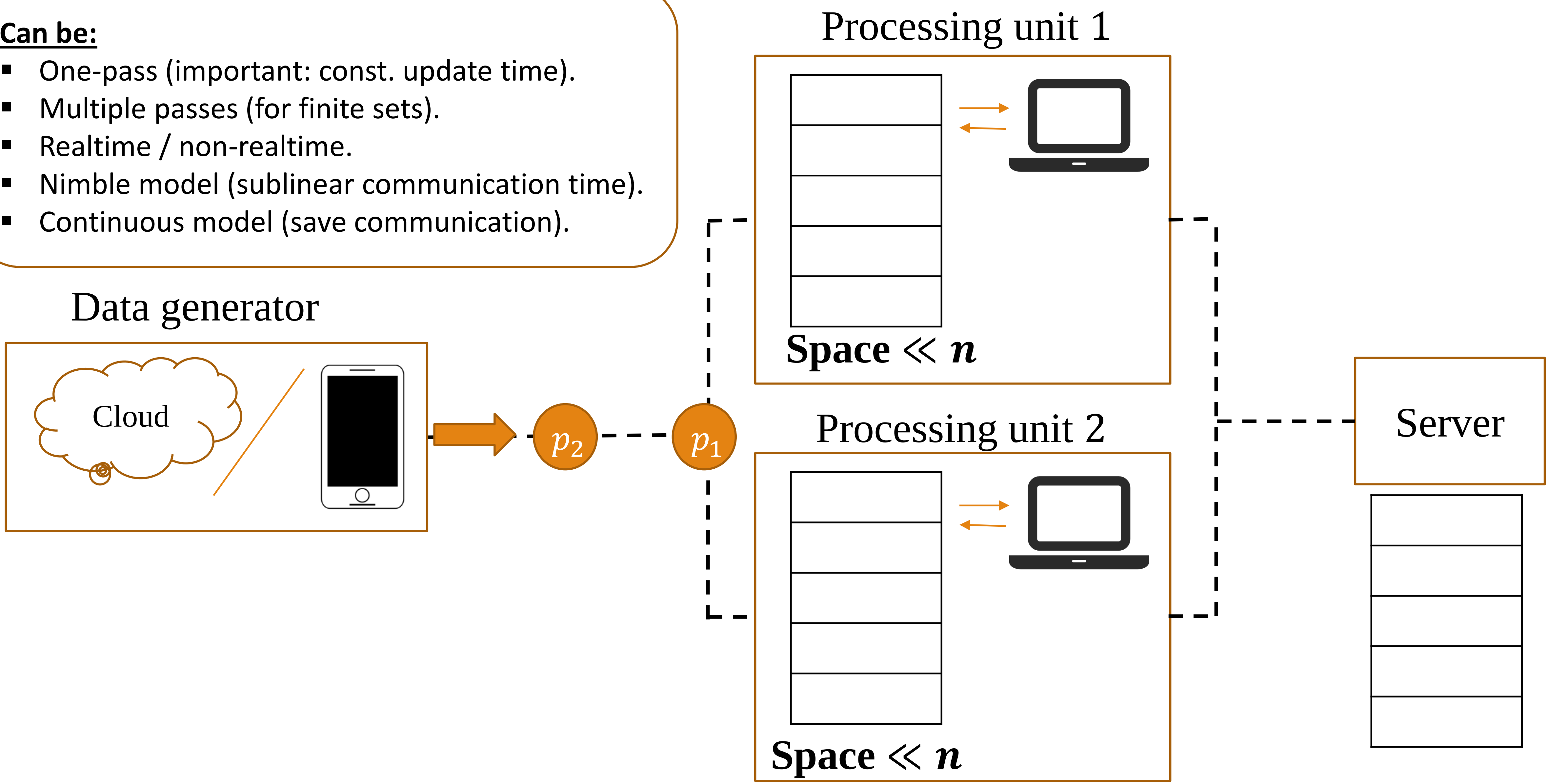
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

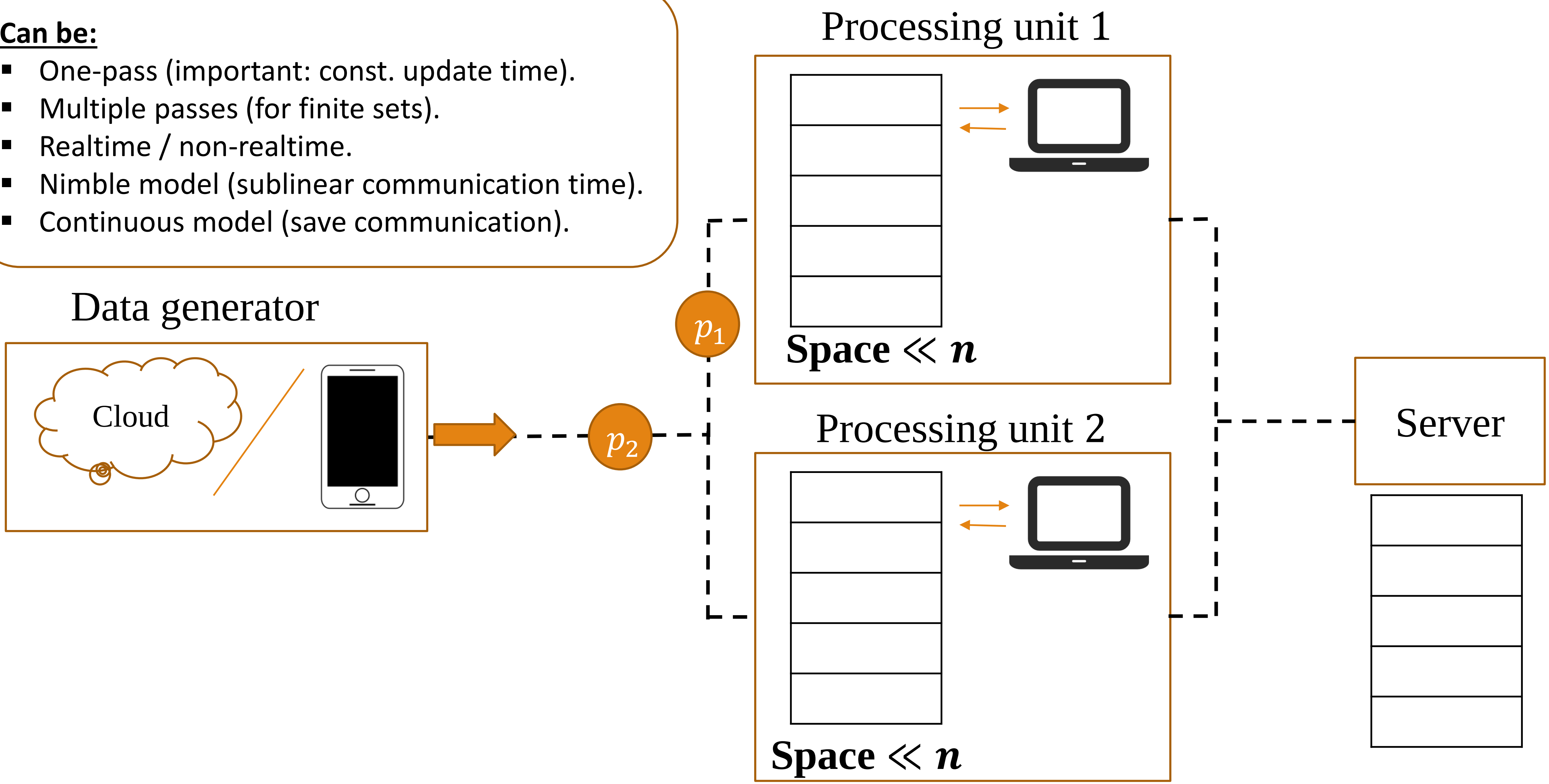
Can be:

- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

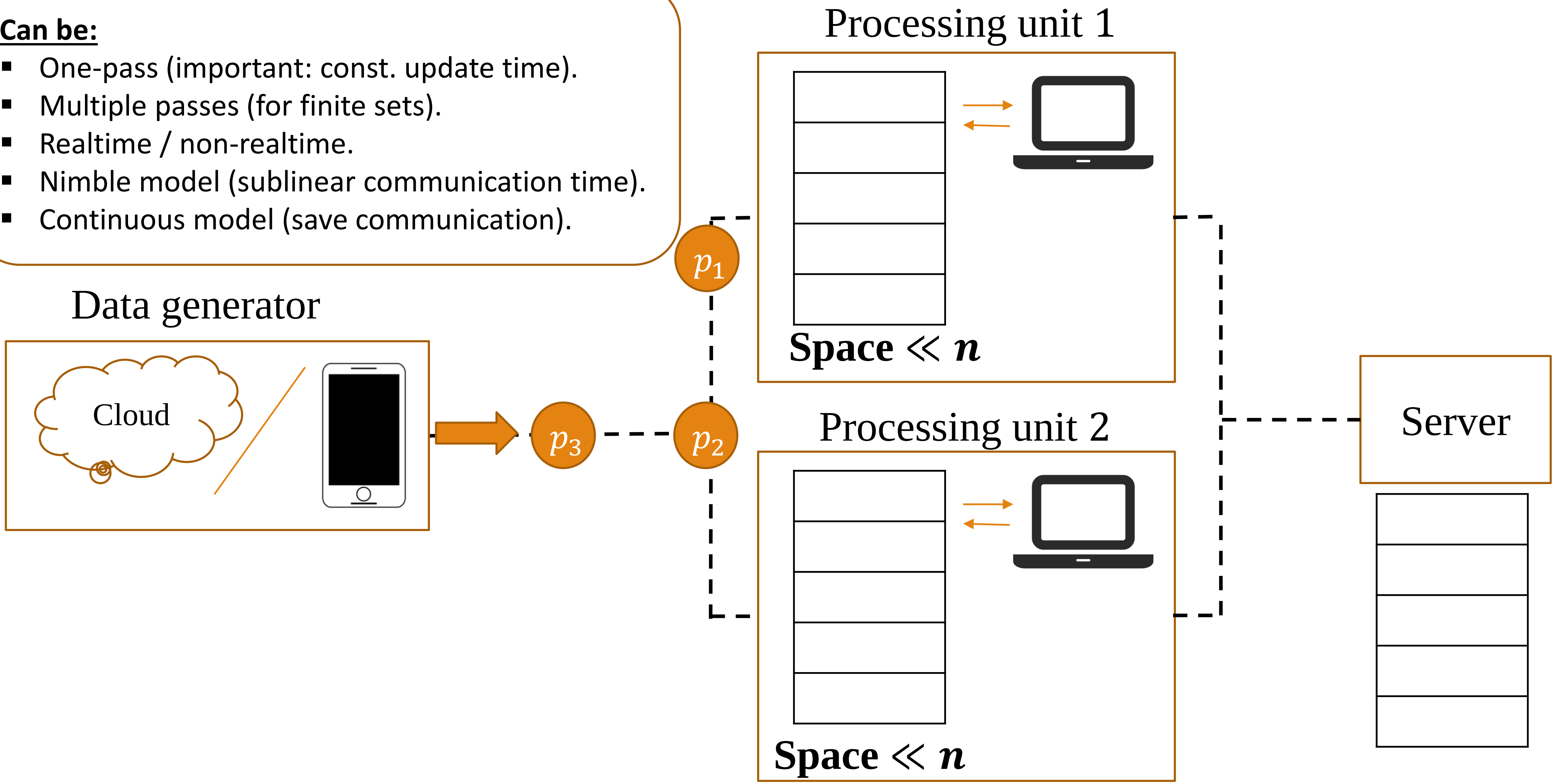
Can be:
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

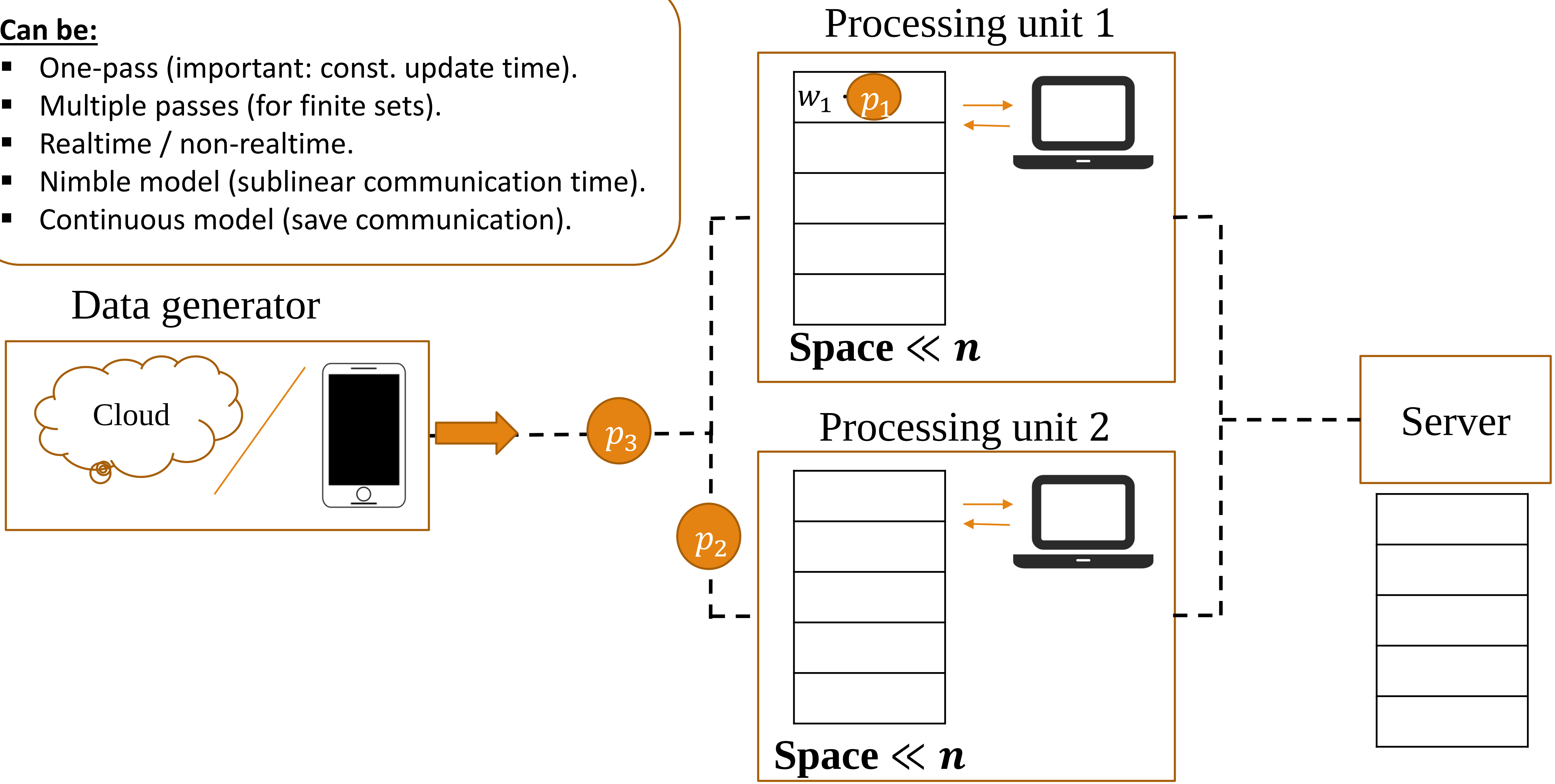
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

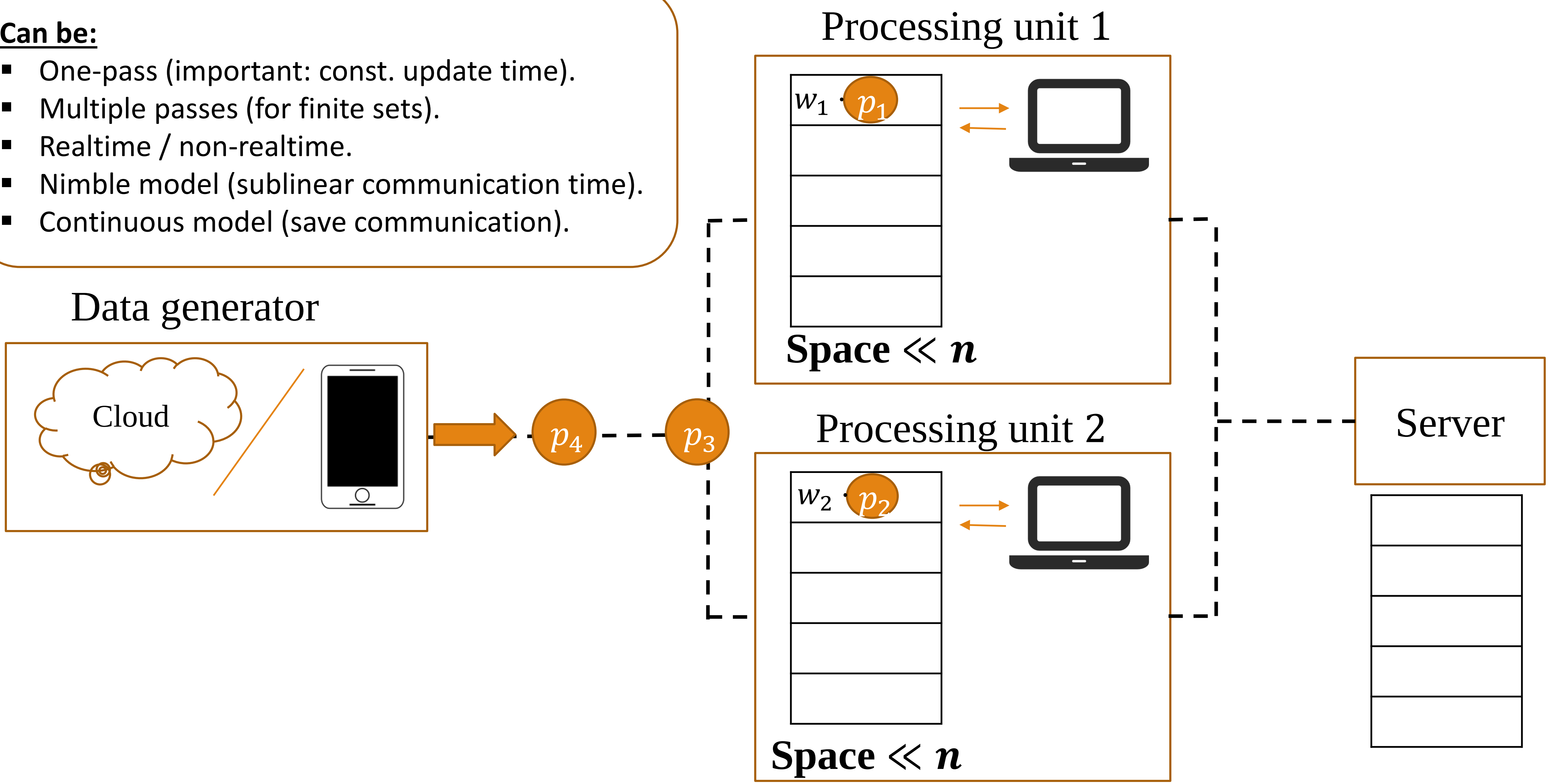
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

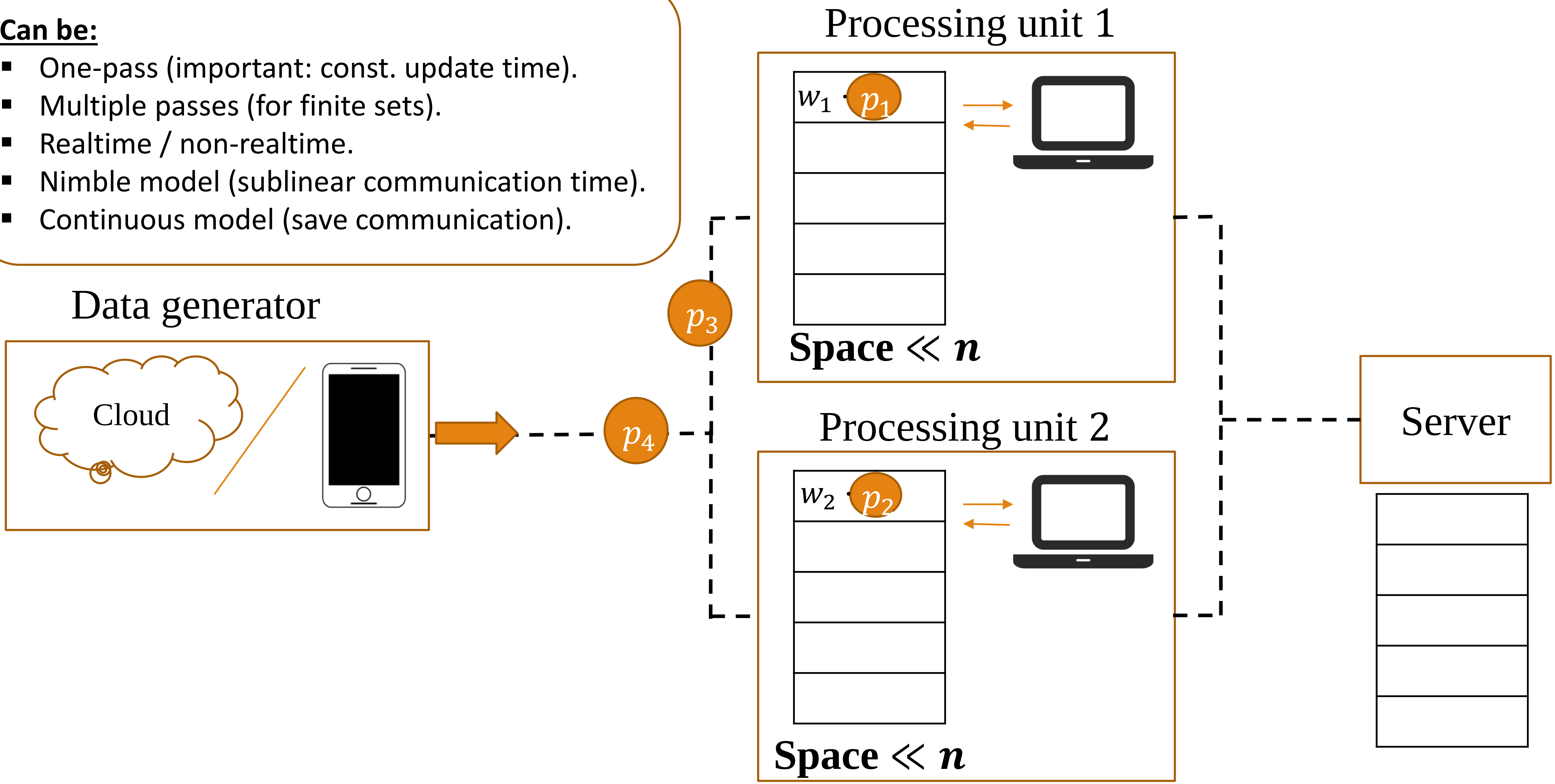
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

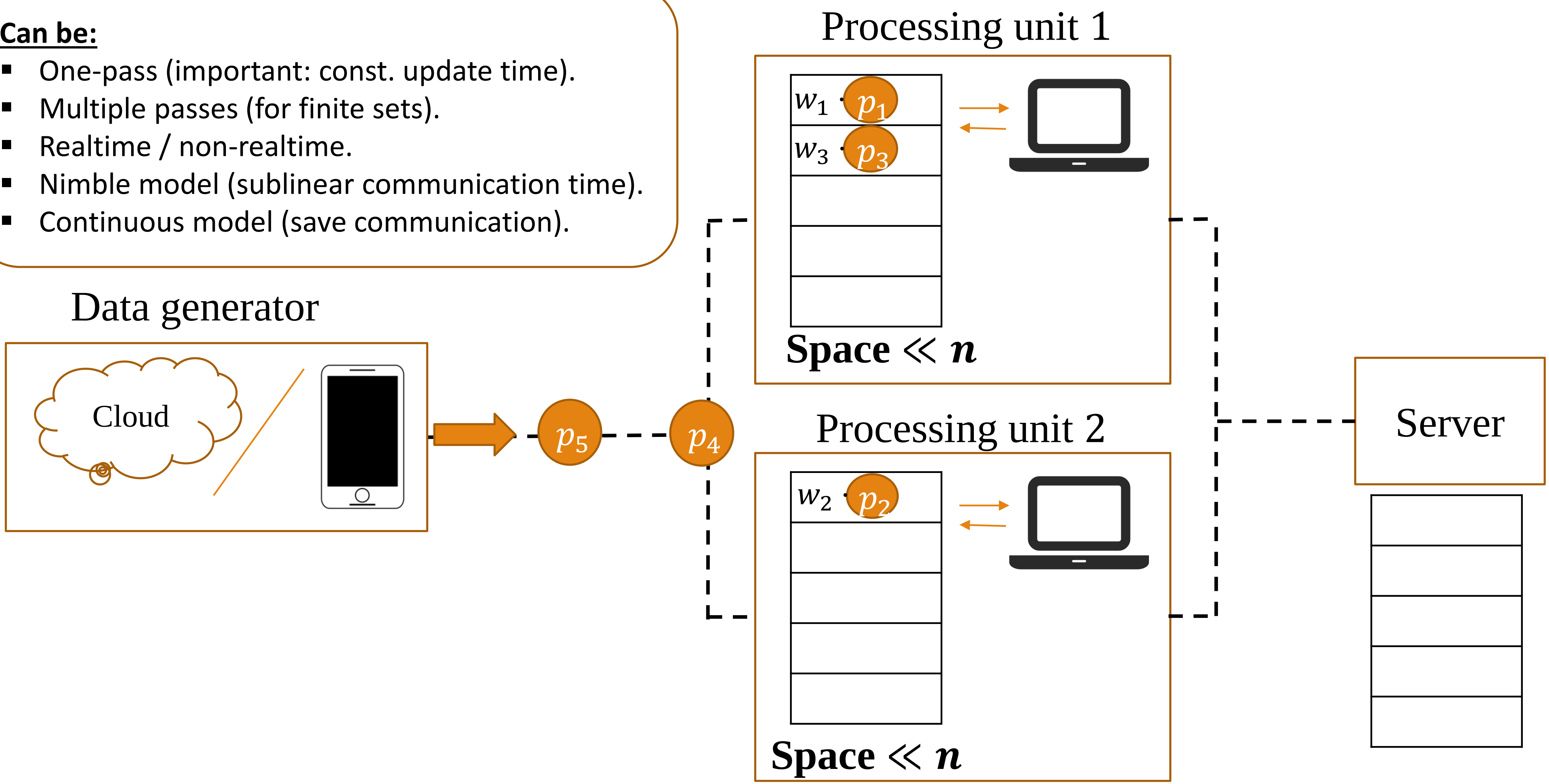
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

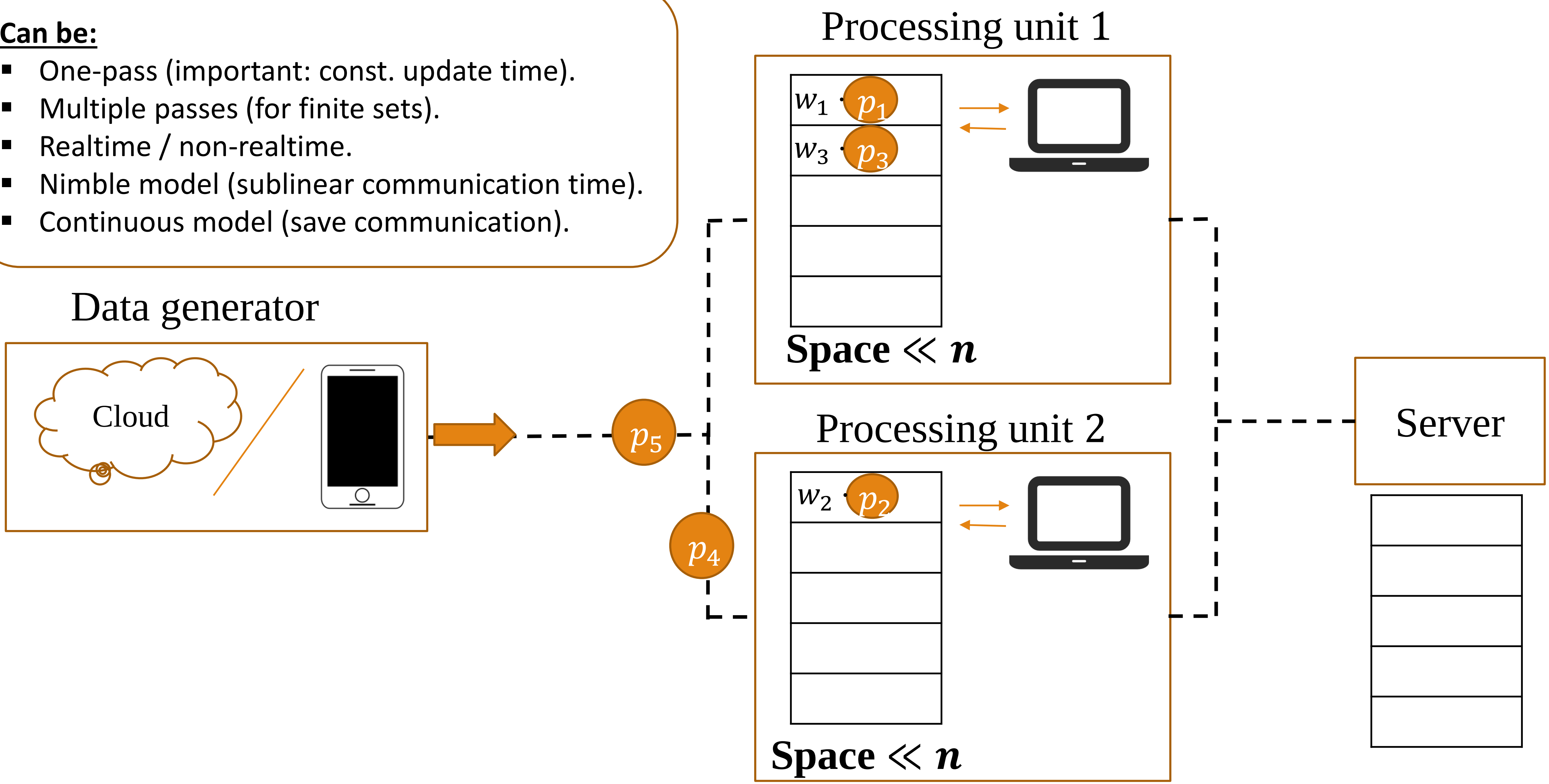
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

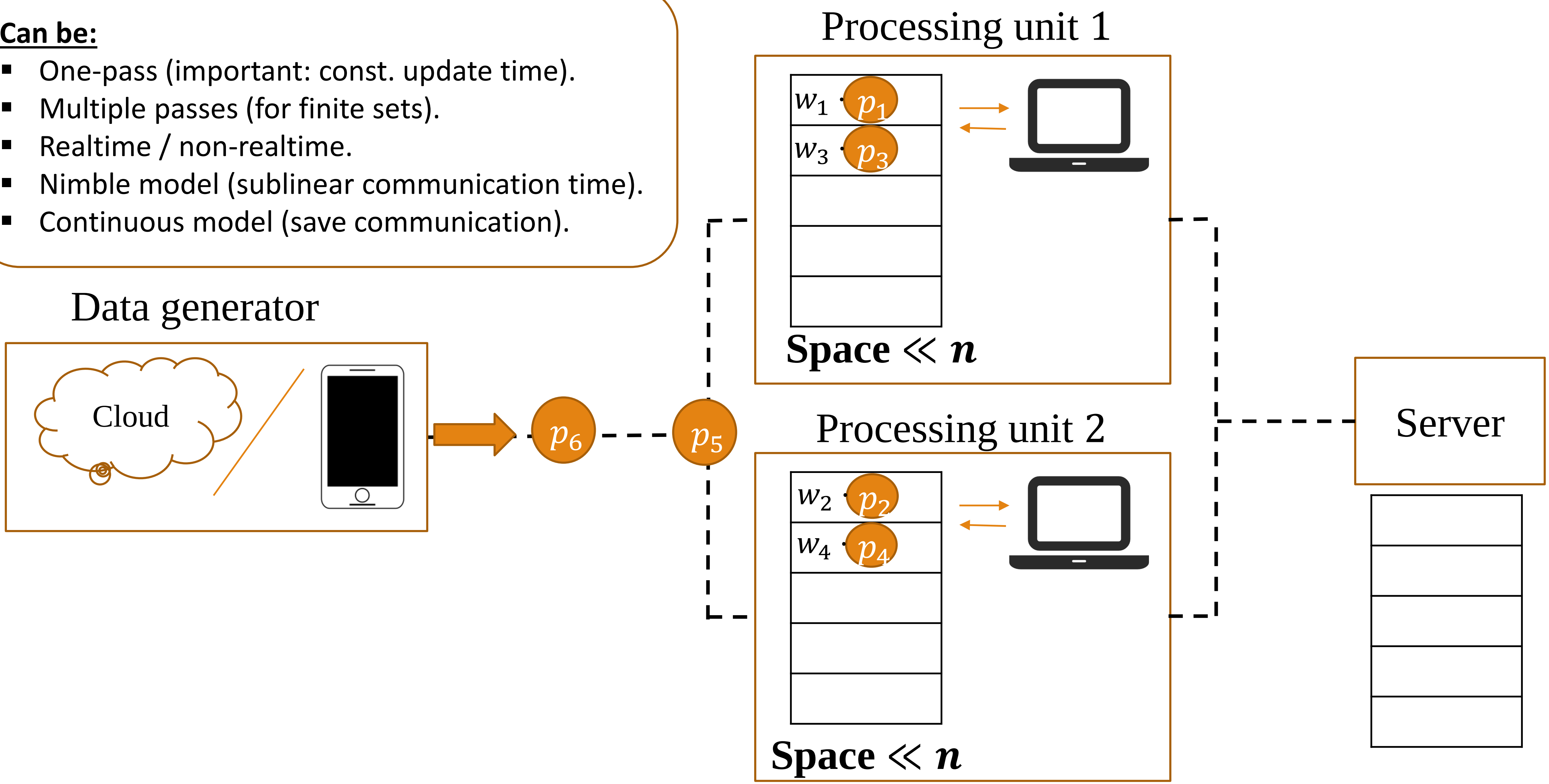
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

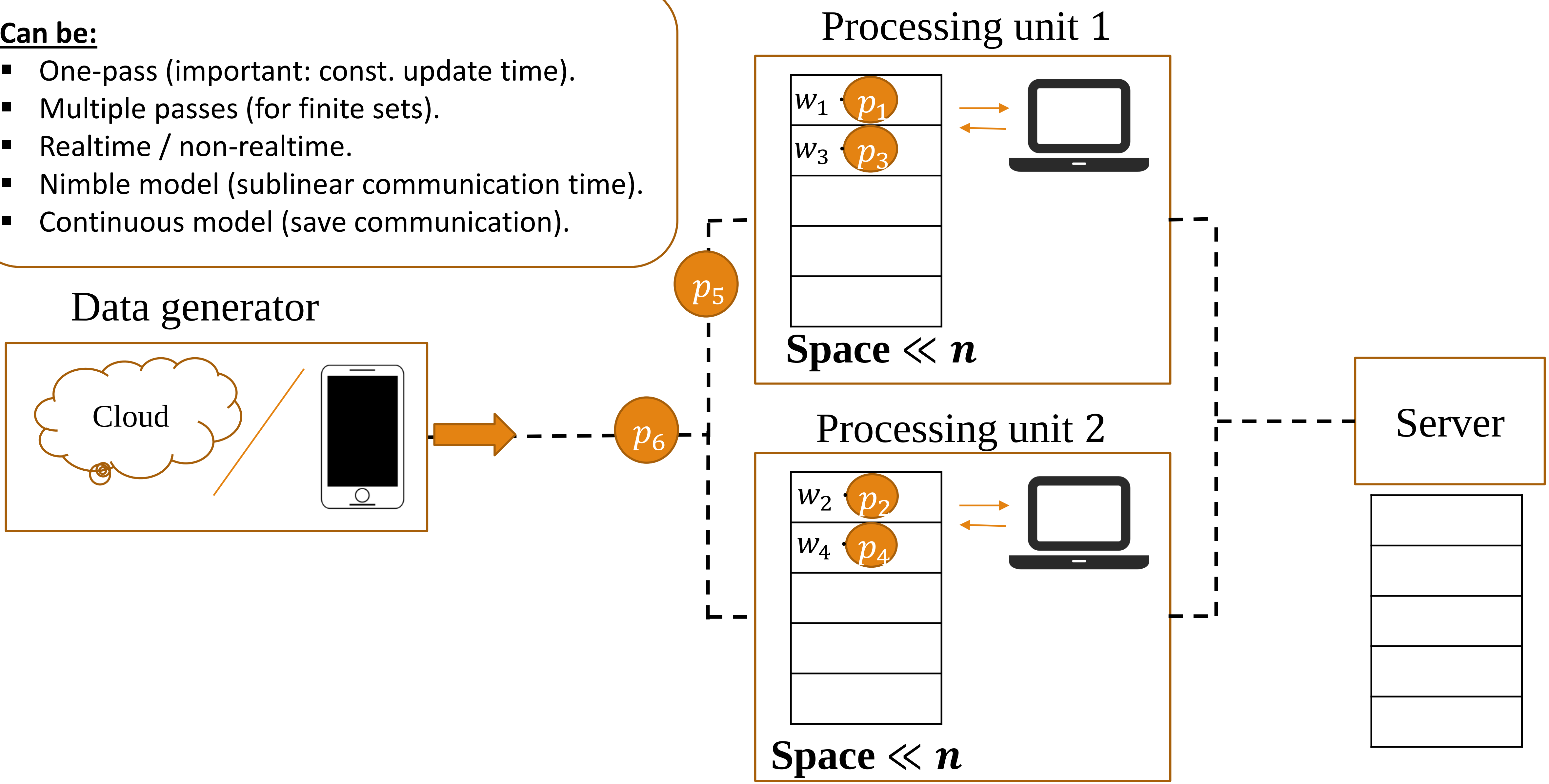
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

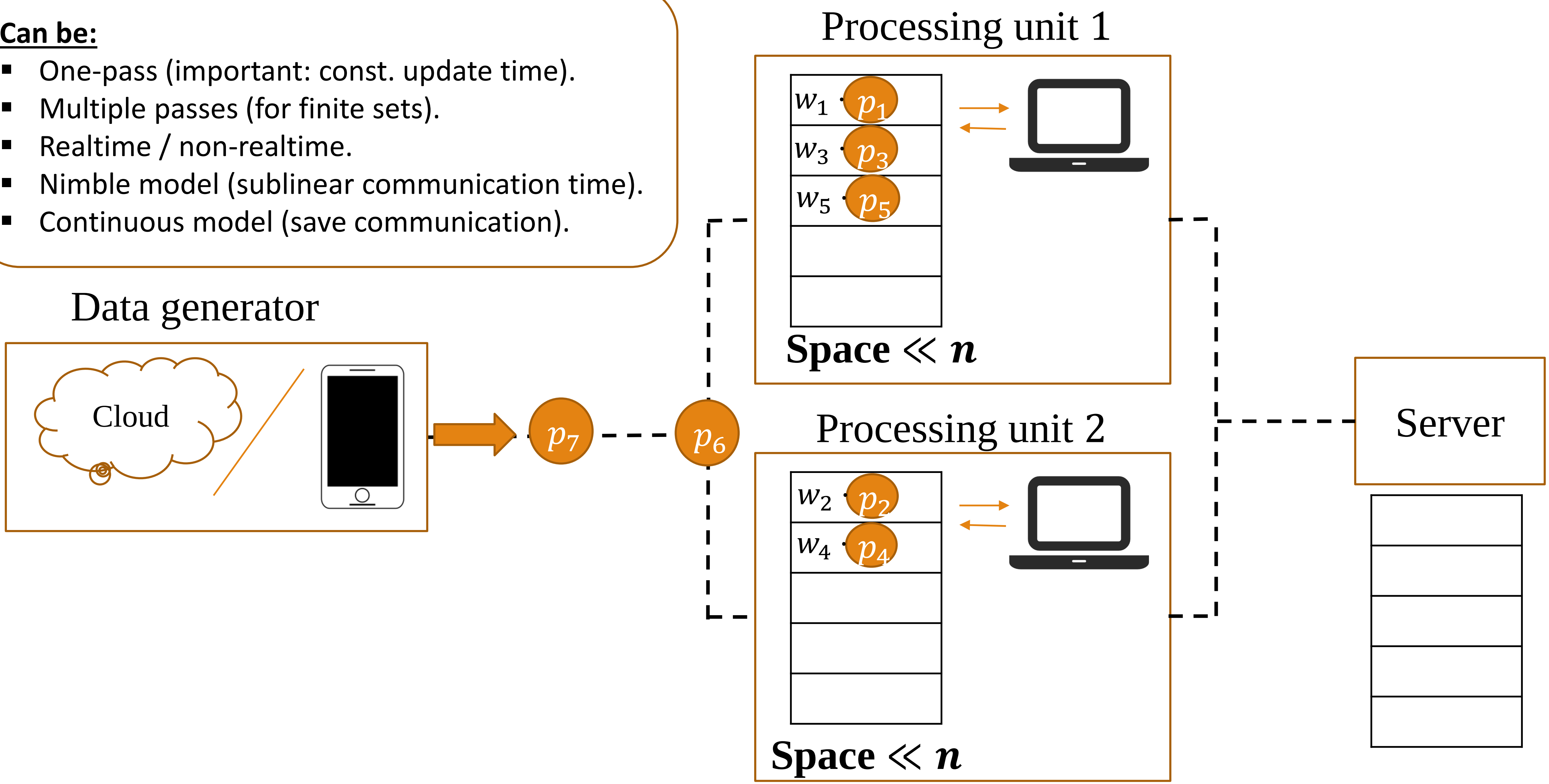
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

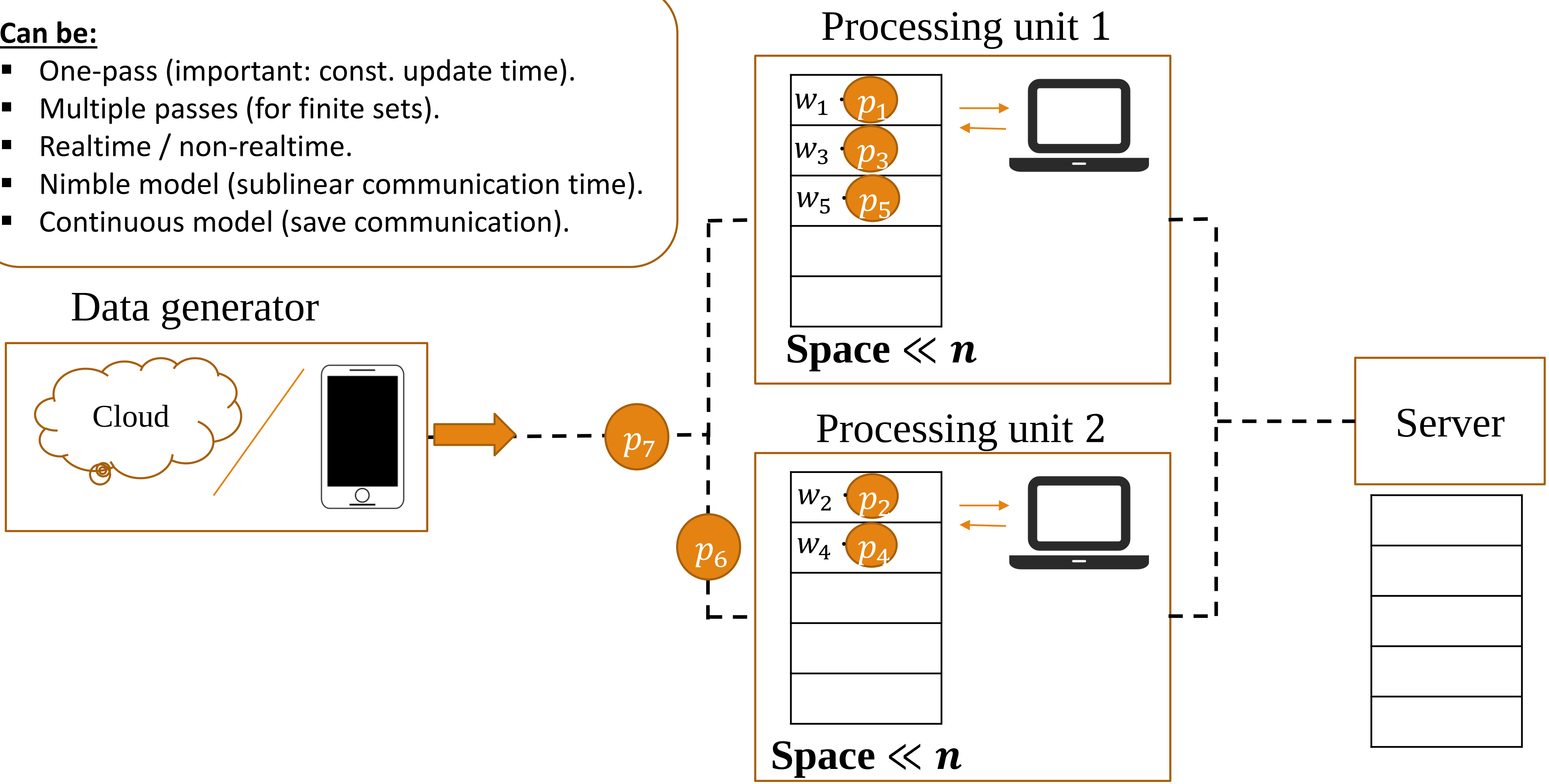
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

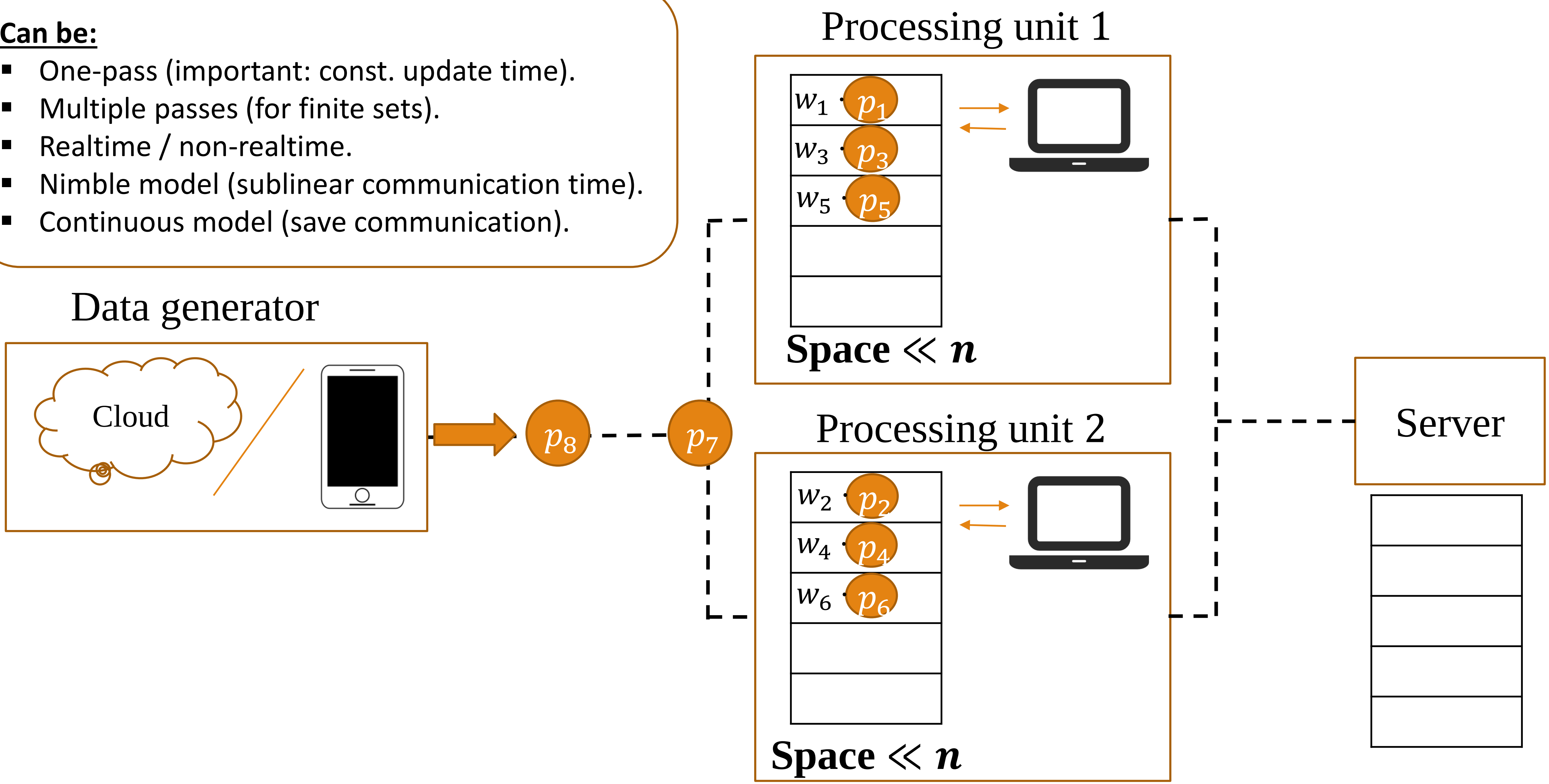
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

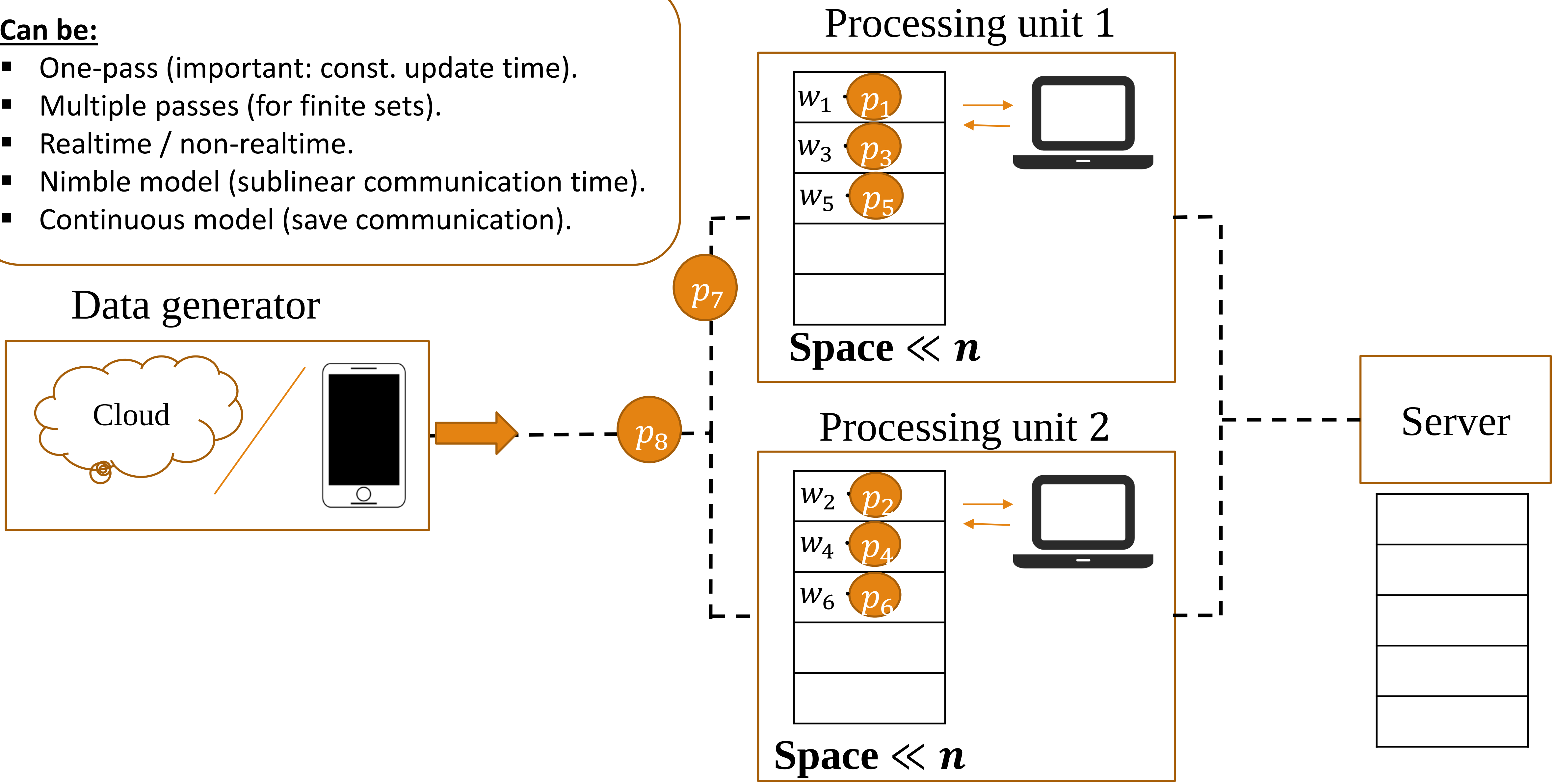
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

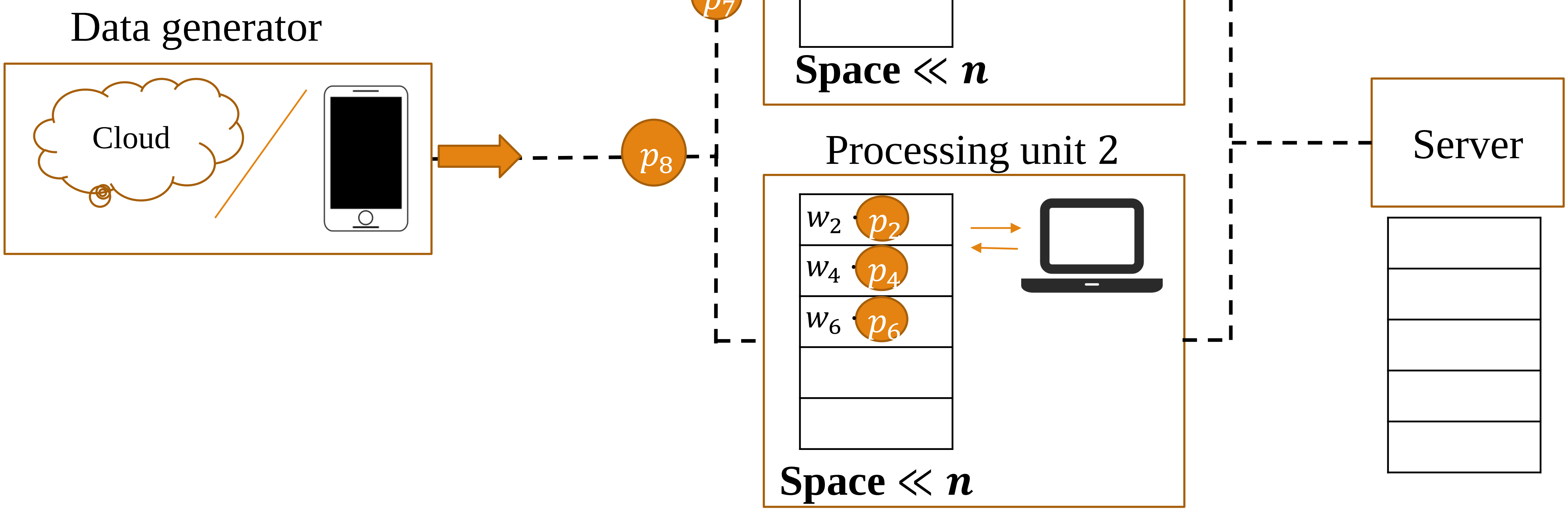
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

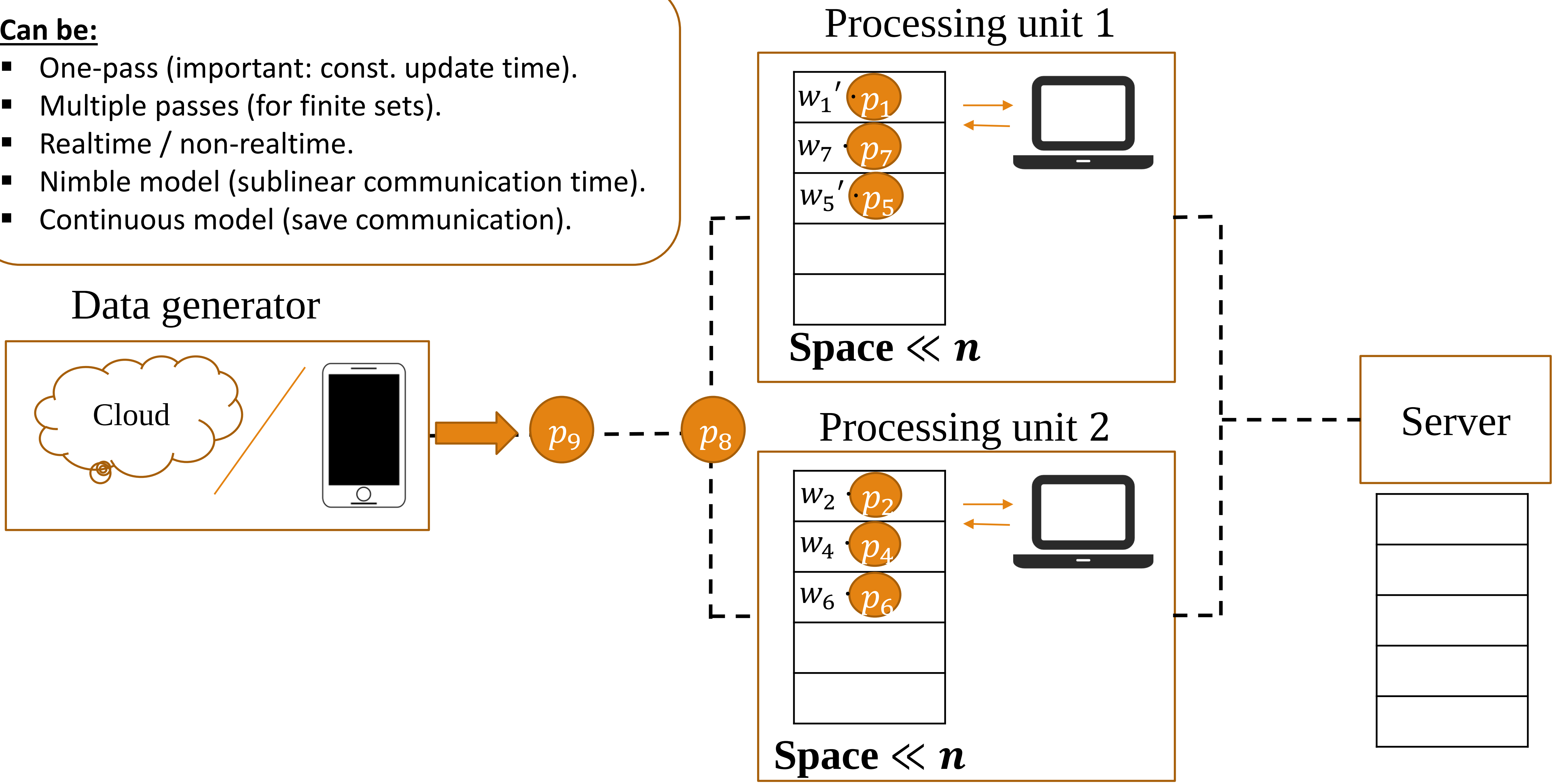
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

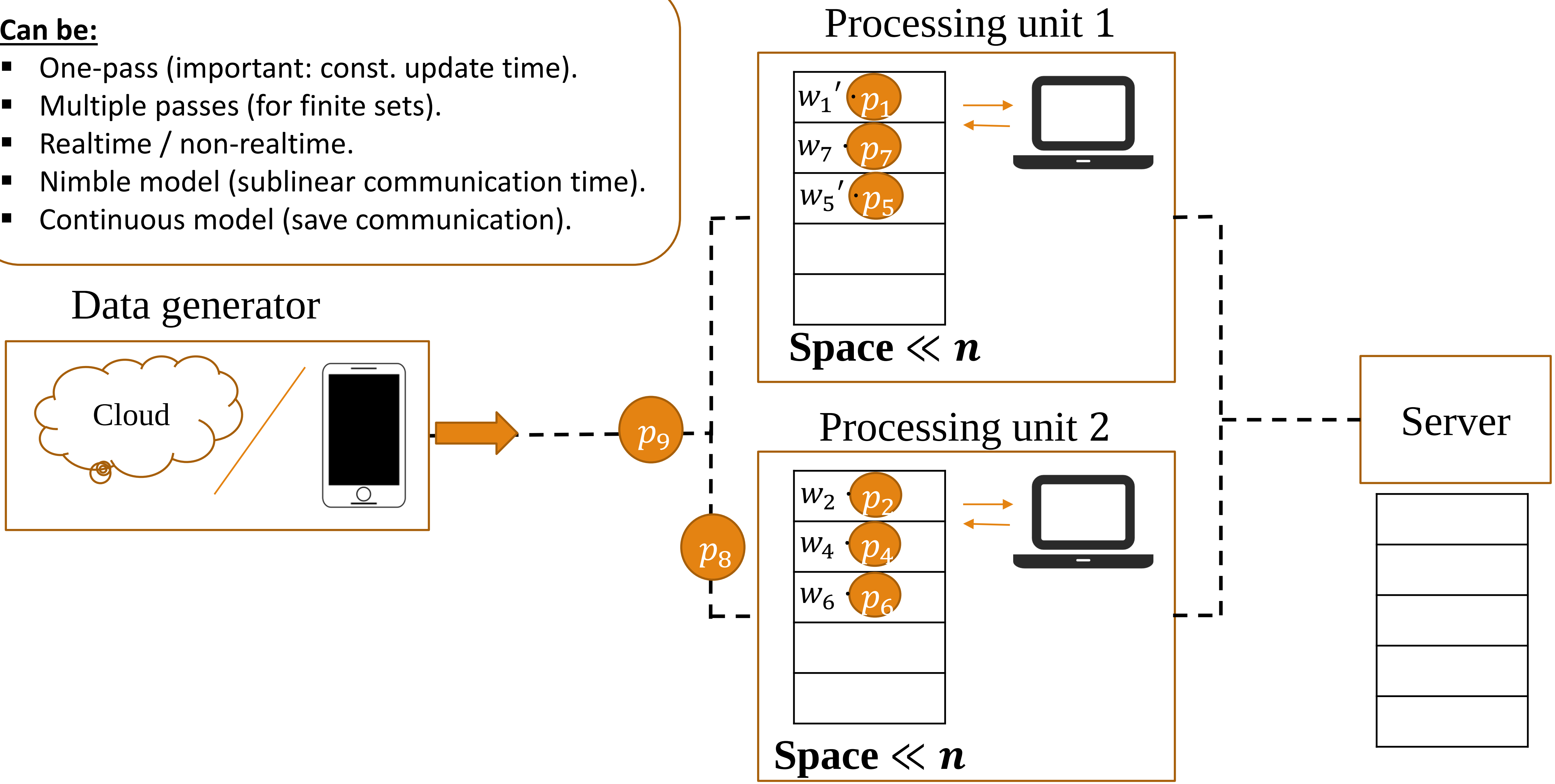
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

Can be:

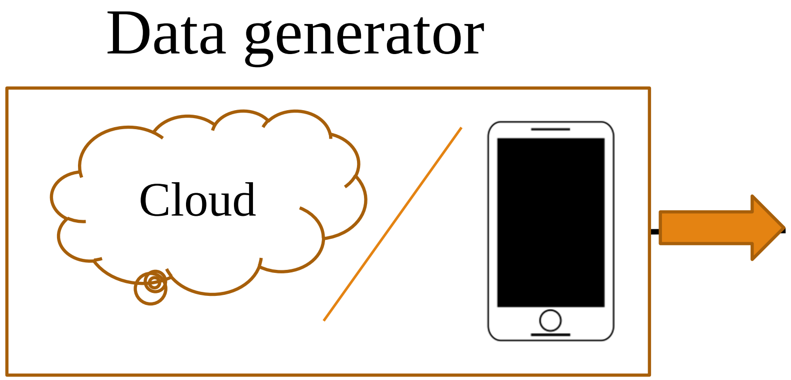
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).

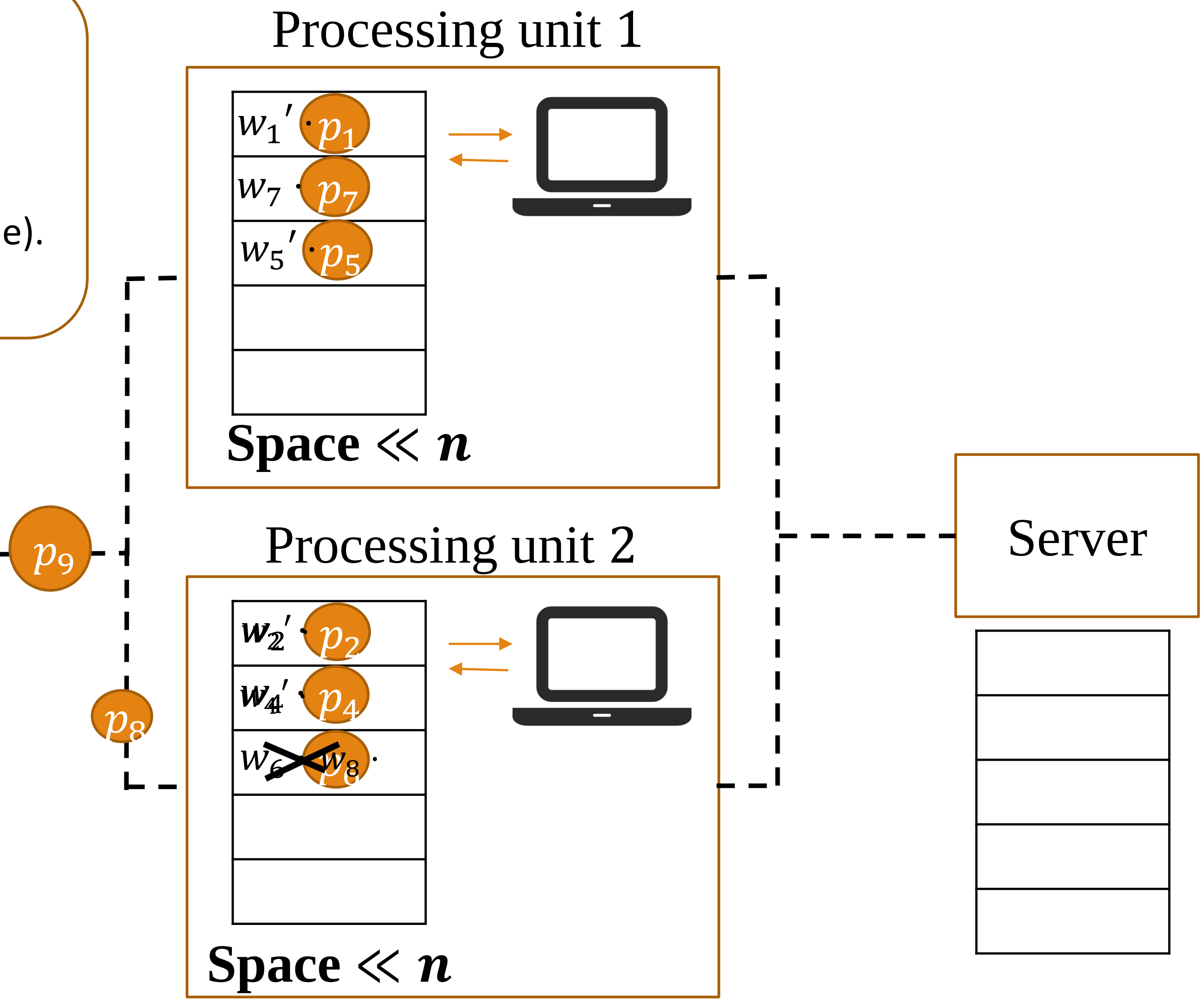


Streaming and distributed algorithms:

Can be:

- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).

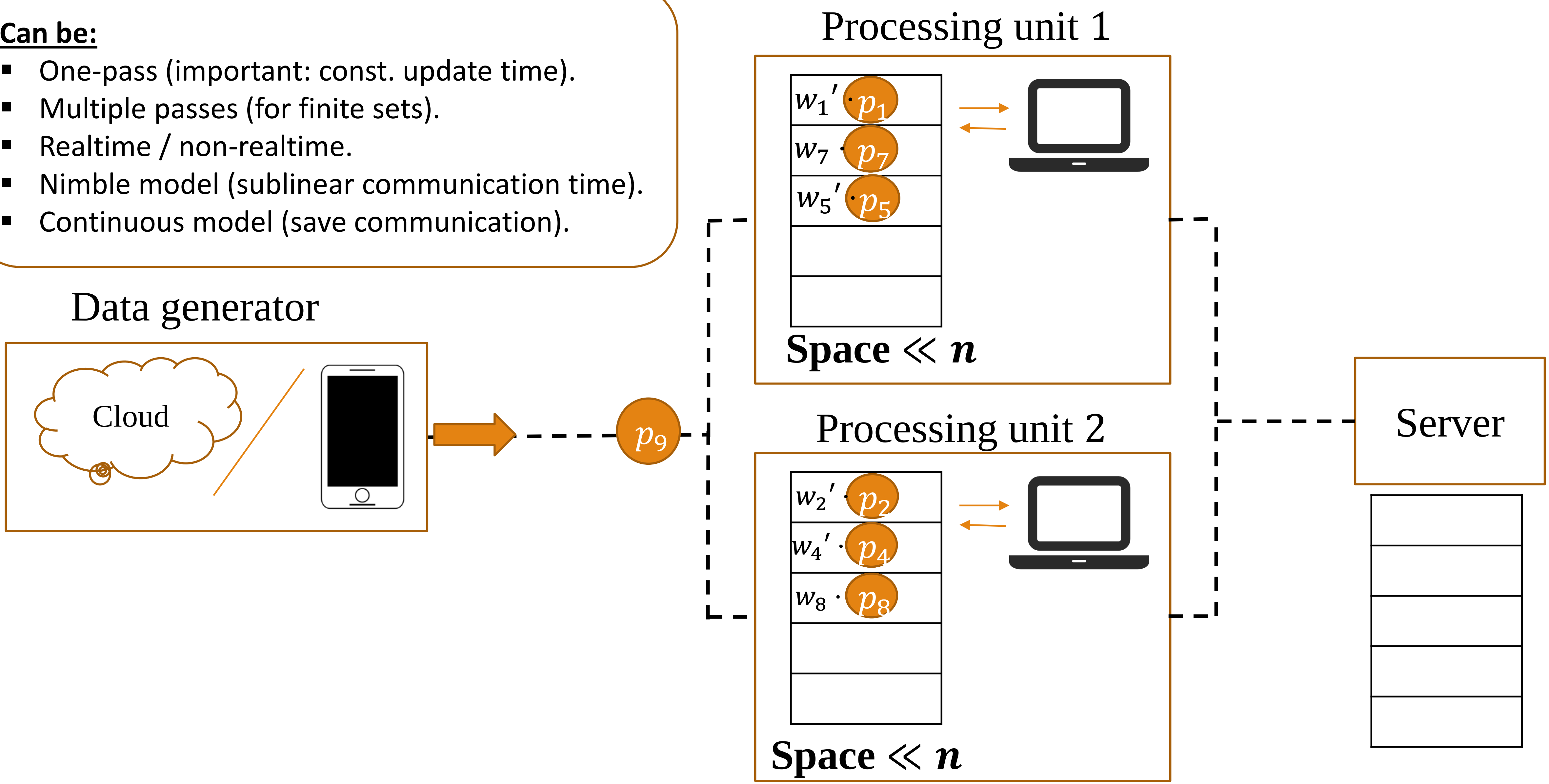




Streaming and distributed algorithms:

Can be:

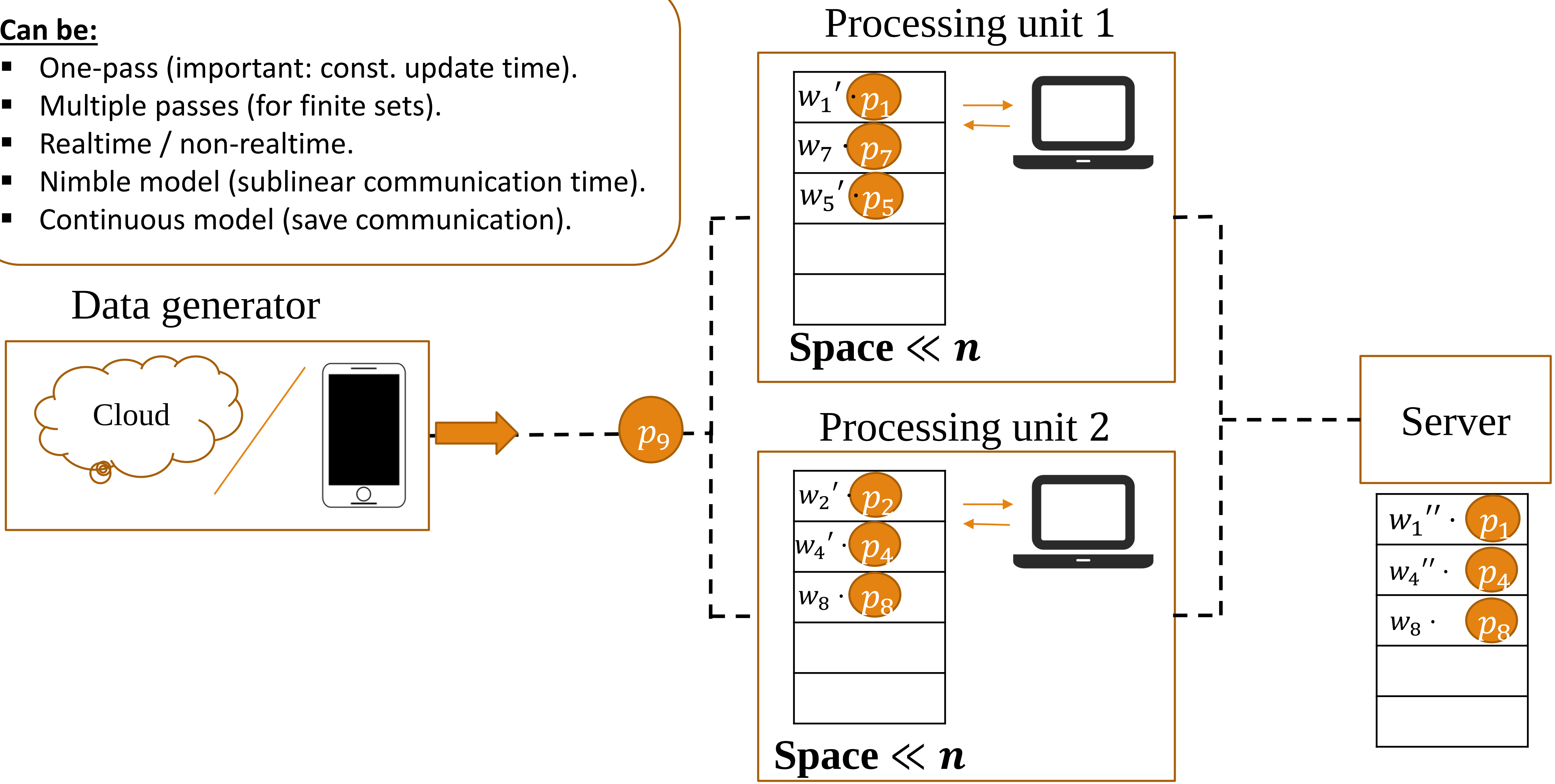
- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



Streaming and distributed algorithms:

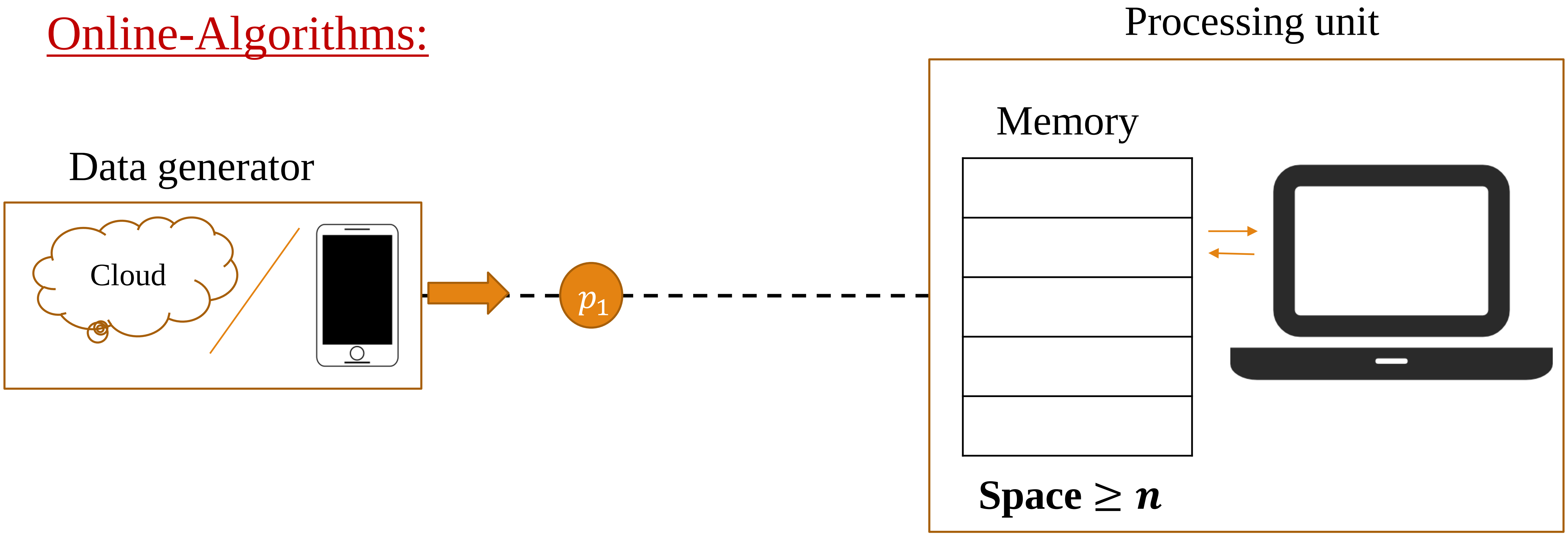
Can be:

- One-pass (important: const. update time).
- Multiple passes (for finite sets).
- Realtime / non-realtime.
- Nimble model (sublinear communication time).
- Continuous model (save communication).



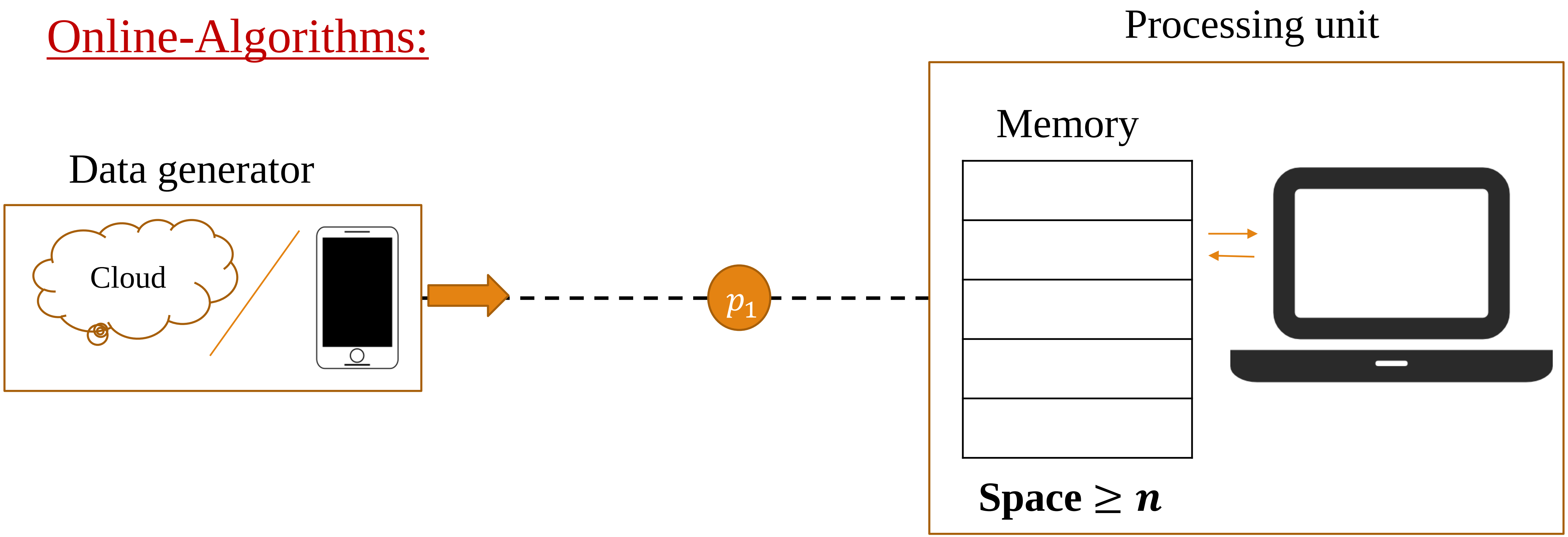
Big data models

<u>Online-Algorithms:</u>



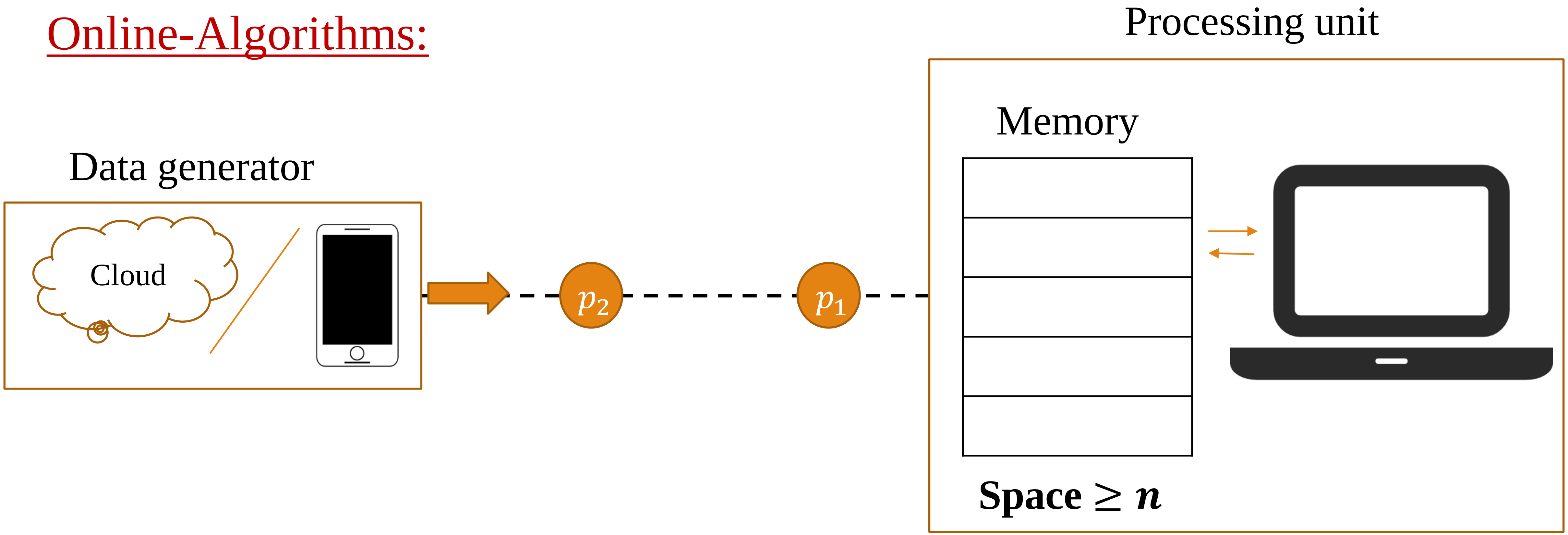
Big data models

Online-Algorithms:



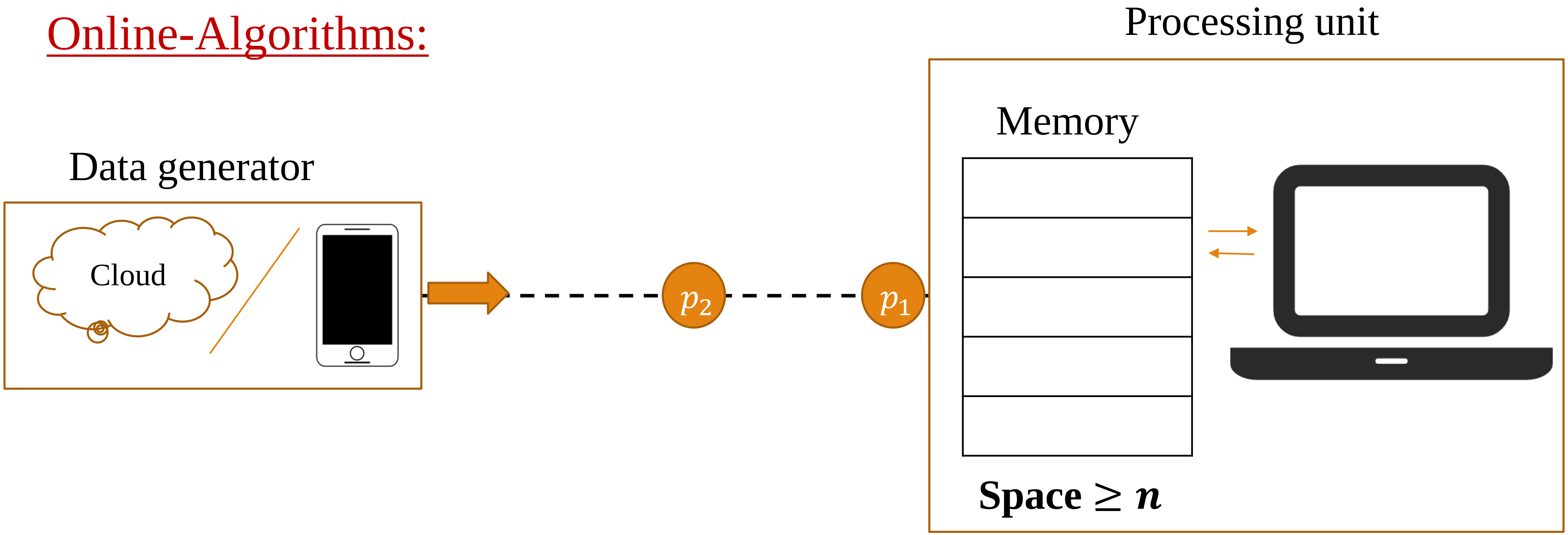
Big data models

Online-Algorithms:



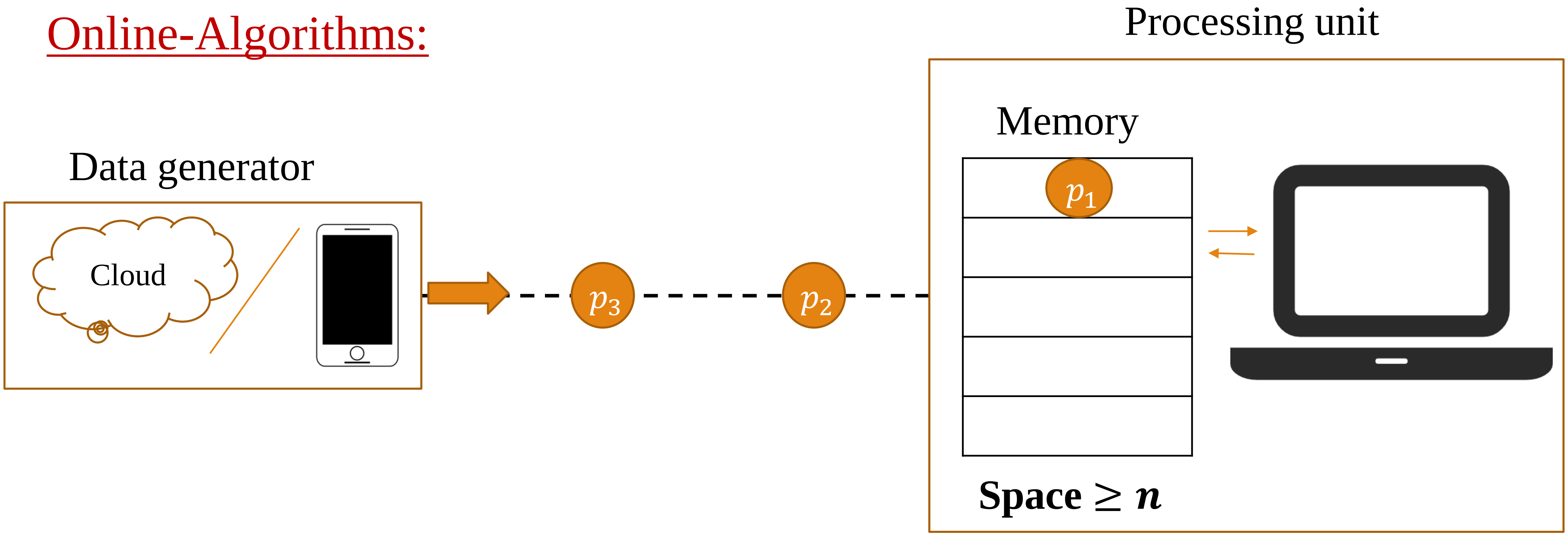
Big data models

<u>Online-Algorithms:</u>



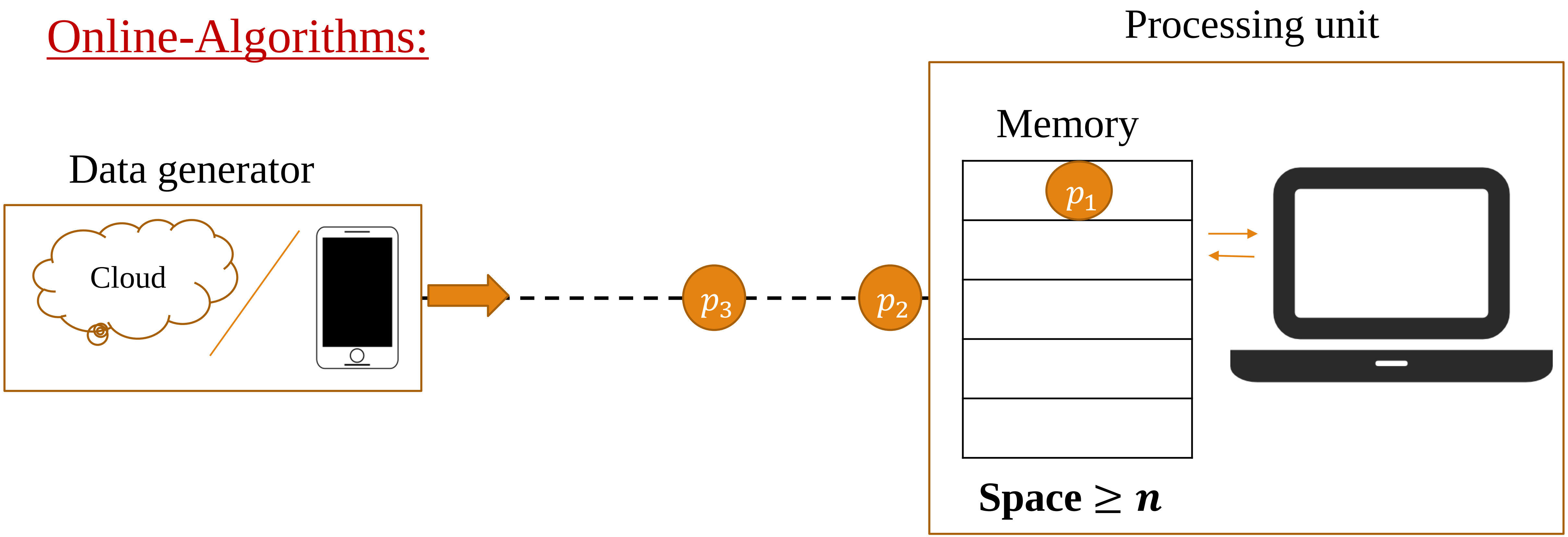
Big data models

Online-Algorithms:



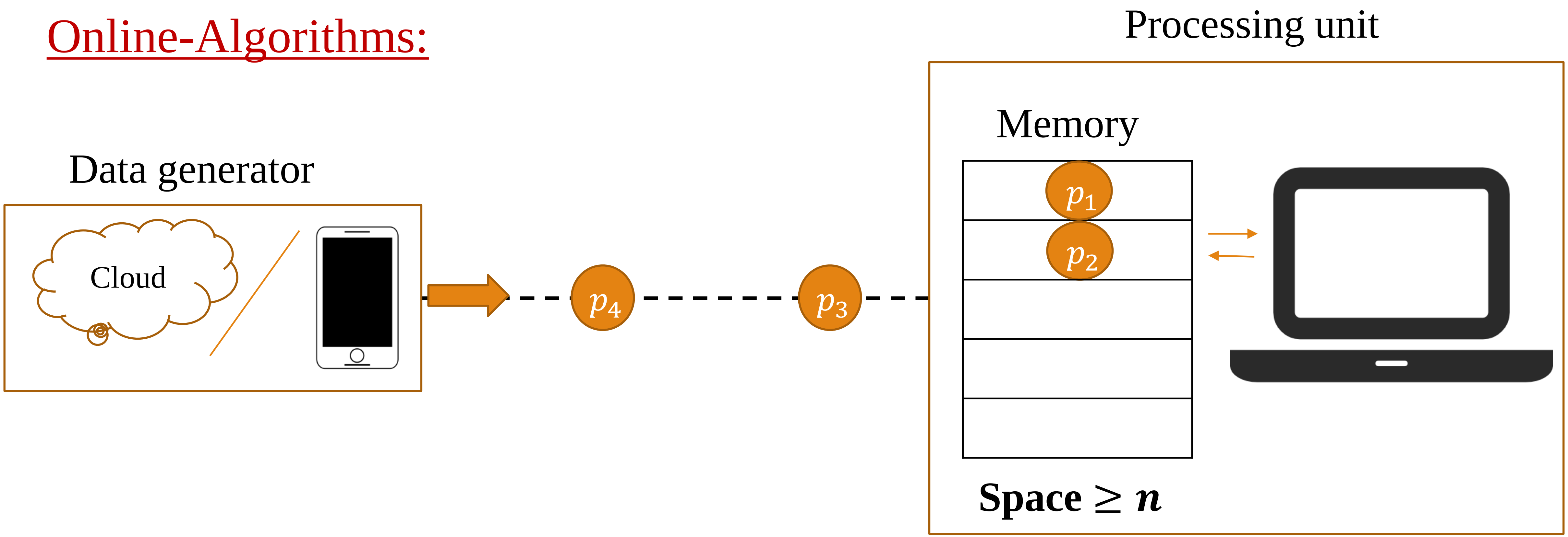
Big data models

Online-Algorithms:



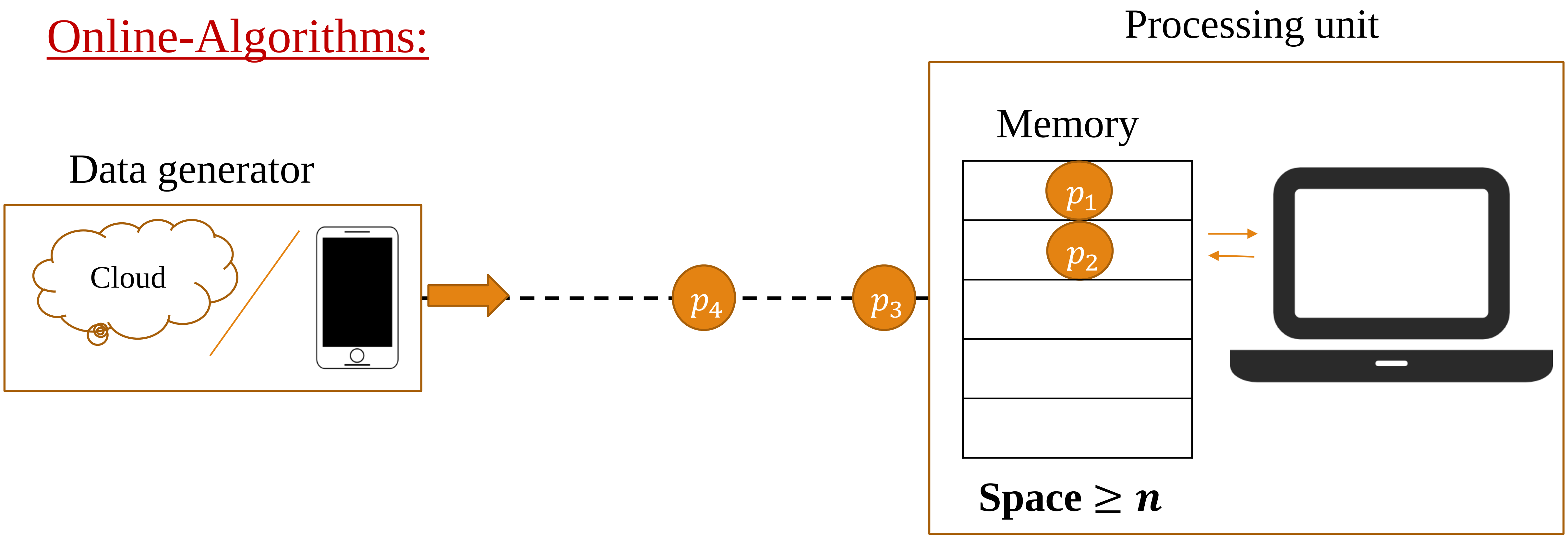
Big data models

Online-Algorithms:



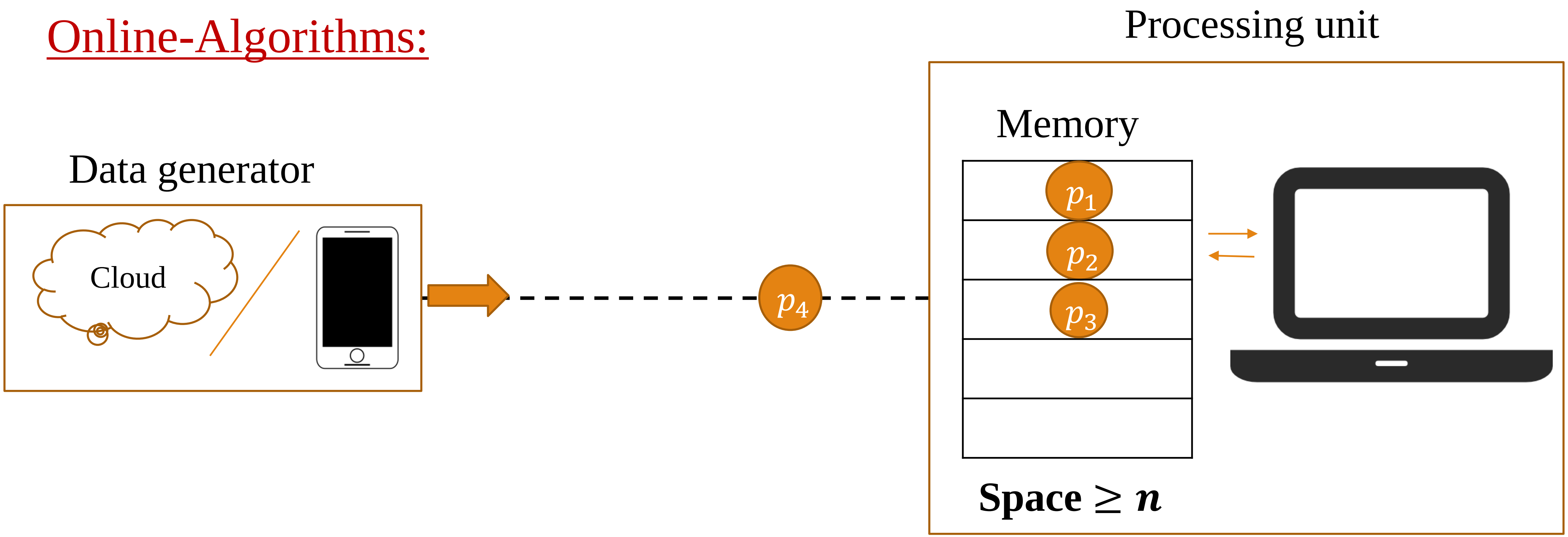
Big data models

Online-Algorithms:



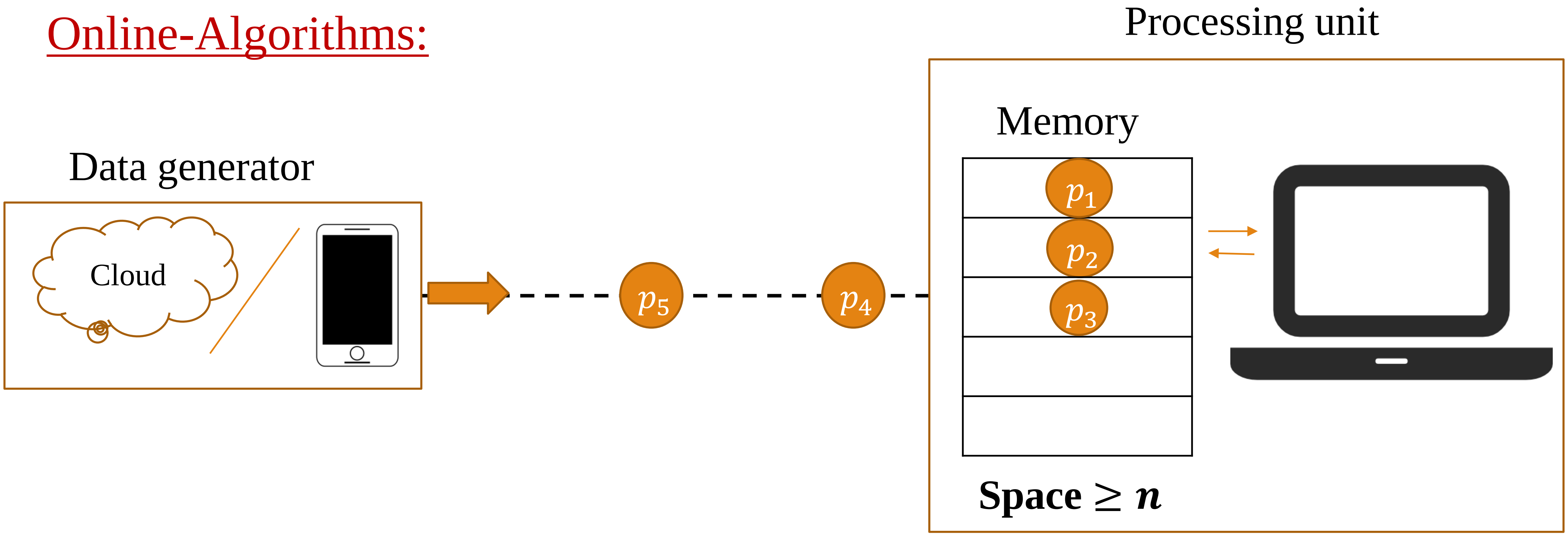
Big data models

Online-Algorithms:



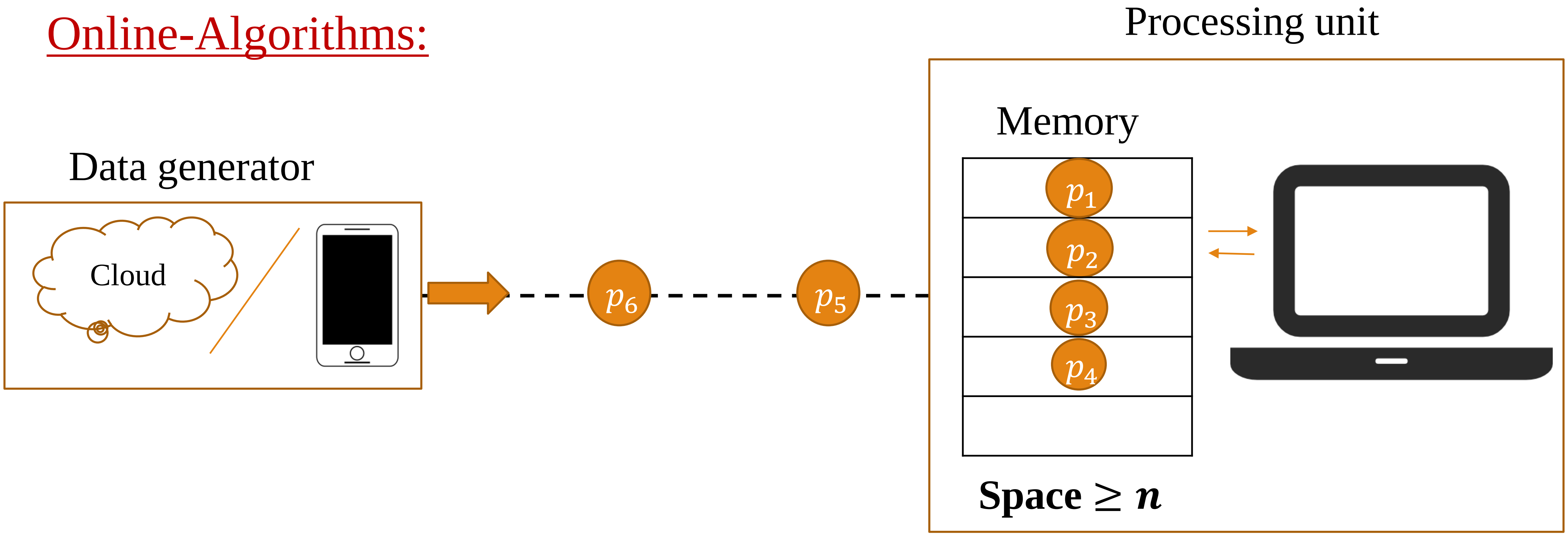
Big data models

Online-Algorithms:



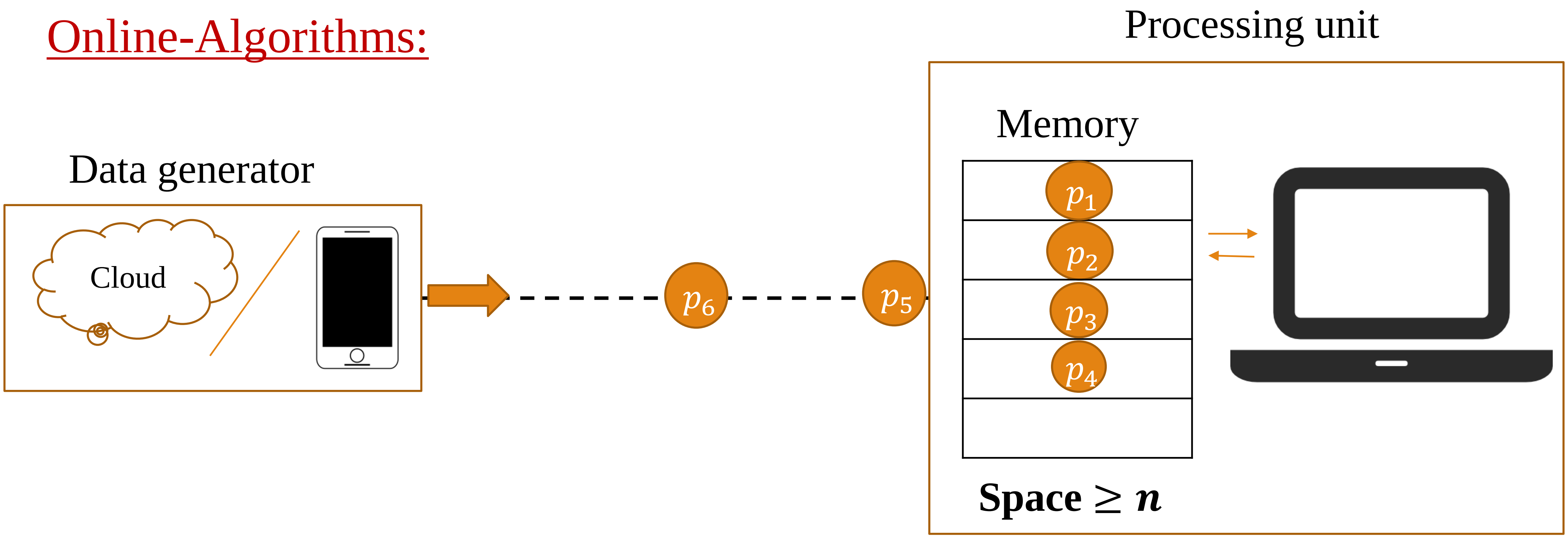
Big data models

Online-Algorithms:



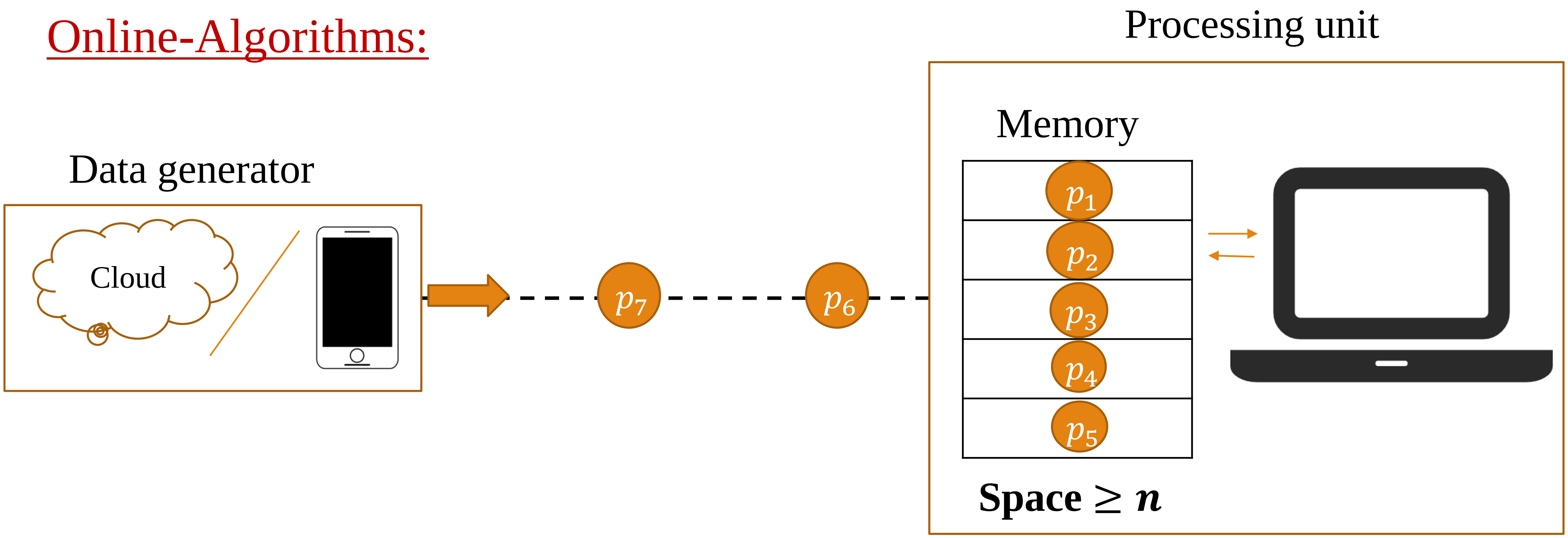
Big data models

Online-Algorithms:



Big data models

Online-Algorithms:



Processing units

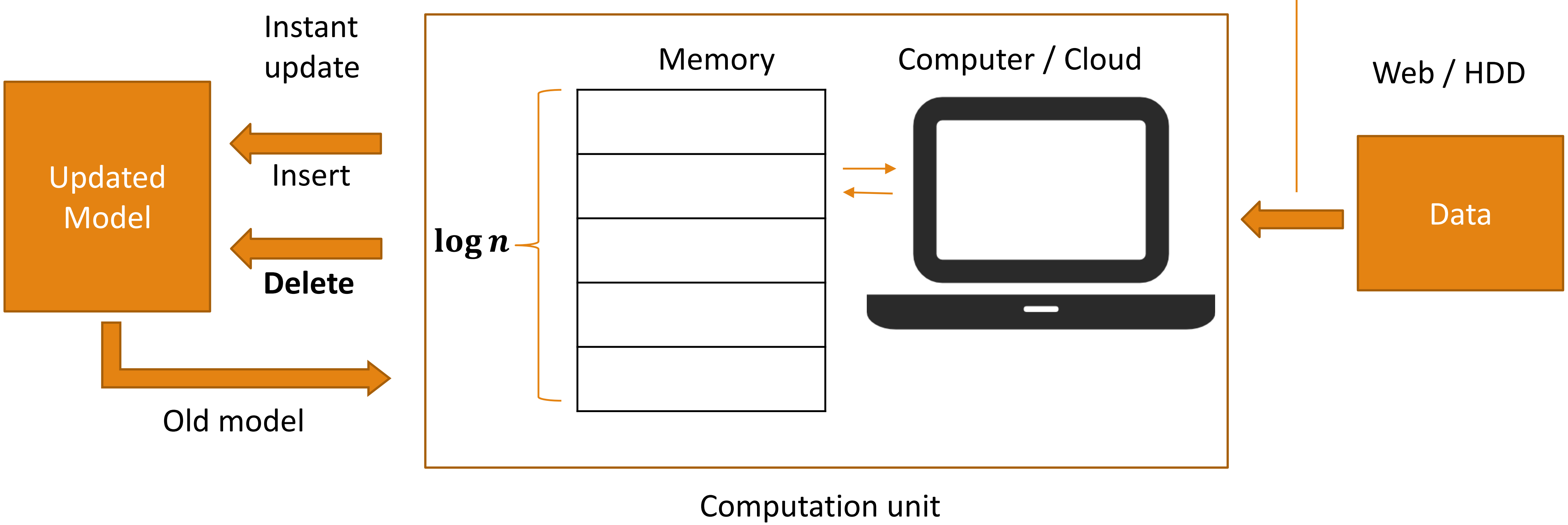
Can be:

- Threads
  - Shard memory (centralized)
- Machines (network, cloud)
  - Distributed Data
  - Possible Graph of Communication
- GPU (limited parallel local computations)
- IoT – low energy and weak computations: Arduino, Rpie
  - Sensors that collect a lot of data, usually to the web
- Real-Time: Face recognition (sec), Quadcopters (msec), Video Stre

Computation models

- Off-line
- Streaming (insertions only)
- Dynamic Data (+ deletions)
    - Sliding window
- Turn-style model (coordinates updates)
- Kinematic data (moving points)

Dynamic-Model:



Motivation

New computation models

- Big Data
- Streaming real-time data
- Distributed data

Limited hardware

- Computation: IoT, GPU
- Energy: smartphones, drones

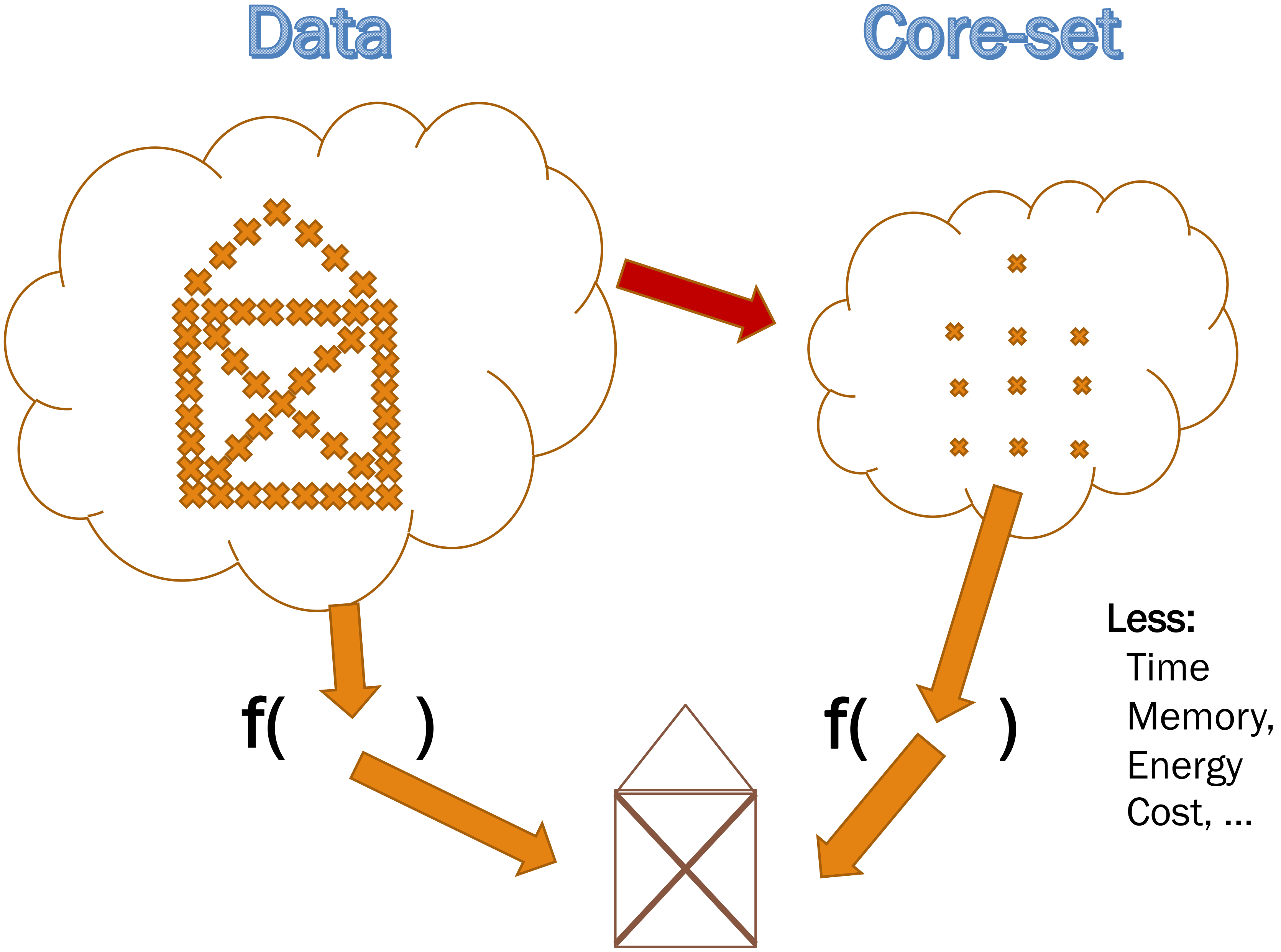
Common solution:

- New optimization algorithms

How to handle all these new computation models?

- Possible approach:
  - design new learning/optimization algorithms under the new constraints
- In this class:
  - Use data summarization/reduction (called core-set)
  - Solve problem on coreset using existing algorithms

Data summarization via Core-sets



Query Space

Definition: Let

- P be a set of n elements.
- Q be a set of possibly infinite elements / queries.
- $\omega: P \rightarrow [0, \infty)$.
- $f: P \times Q \rightarrow \mathbb{R}$ be a cost function.

The tuple (P, ω, Q, f) is called a query space.

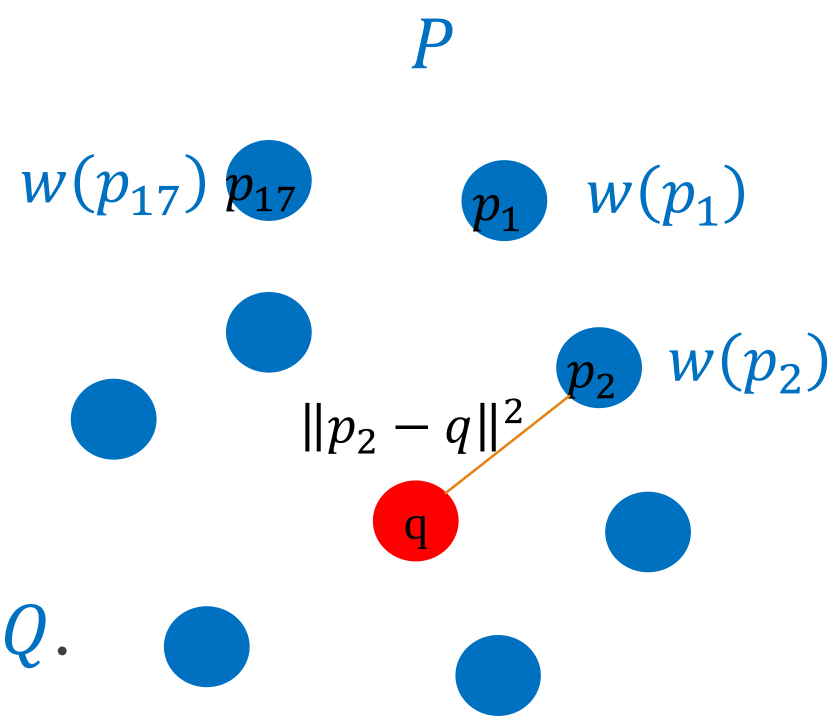
- The cost of a query $q \in Q$ is defined by

$$\bar{f}(P, \omega, q) := \sum_{p \in P} \omega(p) \cdot f(p, q)$$

Query Space

Problem: One mean

- $P = \{p_1, \cdots, p_n\} \subseteq \mathbb{R}^d$.
- $Q = \mathbb{R}^d$ (every possible point in $\mathbb{R}^d$).
- $\omega: P \to [0, \infty)$.
- $f(p, q) = ||p - q||^2$ for every $p \in P$ and $q \in Q$.



The tuple (P, ω, Q, f) is our query space.

Query Space

Problem: Points to Hyperplanes

- $A = \begin{bmatrix} -a_1 - \\ \vdots \\ -a_n - \end{bmatrix} \in R^{n \times d}$ (n points in R^d)
- $S = R^d$ (The normals of hyperplanes in R^d)
- $\omega(\mathrm{a}) = 1$ for every $a \in A$.
- $f(a, x) = |ax|^2$ for every $a \in A$ and $x \in S$.

$$\rightarrow \sum_{i=1}^{n} \omega(a_i) \cdot f(a_i, x) = \|Ax\|^2$$

The tuple $(\mathrm{A}, \omega, \mathrm{S}, f)$ is our query space.

Exact coresets

- Input: Query space (P, w, Q, f)

Exact coreset: (C, μ) is an exact coreset (usually $C \subseteq P$, $\mu: C \to [0, \infty)$), if for every q in Q we have that the sum of the cost function on P with query q is the same as the sum of the cost function on C with query q.

$$\forall \boldsymbol{q} \in \boldsymbol{Q}:$$
$$\sum_{\boldsymbol{p} \in \boldsymbol{P}} \boldsymbol{w}(\boldsymbol{p}) \cdot \boldsymbol{f}(\boldsymbol{p}, \boldsymbol{q}) = \sum_{\boldsymbol{c} \in \boldsymbol{C}} \boldsymbol{\mu}(\boldsymbol{c}) \cdot \boldsymbol{f}(\boldsymbol{c}, \boldsymbol{q})$$

Exact Coreset - Example: Points to Hyperplanes

Reminder: QR Decomposition

Decomposition of $A \in R^{n \times d}$ into $A = QR$ where:

$Q \in R^{n \times d}$ is an orthogonal matrix and $R \in R^{d \times d}$ is an upper triangular matrix.

$$\overbrace{\begin{bmatrix} -a_1- \\ \vdots \\ -a_n- \end{bmatrix}}^{d} = \overbrace{\begin{bmatrix} -e_1- \\ \vdots \\ -e_n- \end{bmatrix}}^{d} \cdot \overbrace{\begin{pmatrix} \langle \mathbf{e}_1, \mathbf{a}_1 \rangle & \langle \mathbf{e}_1, \mathbf{a}_2 \rangle & \langle \mathbf{e}_1, \mathbf{a}_3 \rangle & \dots \\ 0 & \langle \mathbf{e}_2, \mathbf{a}_2 \rangle & \langle \mathbf{e}_2, \mathbf{a}_3 \rangle & \dots \\ 0 & 0 & \langle \mathbf{e}_3, \mathbf{a}_3 \rangle & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}}^{d} \Big\} d$$

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{a}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{a}_2 - \mathrm{proj}_{\mathbf{u}_1}\, \mathbf{a}_2, & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{a}_3 - \mathrm{proj}_{\mathbf{u}_1}\, \mathbf{a}_3 - \mathrm{proj}_{\mathbf{u}_2}\, \mathbf{a}_3, & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
&\vdots & &\vdots \\
\mathbf{u}_k &= \mathbf{a}_k - \sum_{j=1}^{k-1} \mathrm{proj}_{\mathbf{u}_j}\, \mathbf{a}_k, & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}
\end{aligned}$$

Exact Coreset via QR-Decomposition

Input: Points–Hyperplane Query Space: (A, w, S, f).

Goal: Compute $R \in \mathbb{R}^{d\times d}$ such that $f(A,x) = f(R,x)$ **for every** $\boldsymbol{x} \in \boldsymbol{S}$.

Let Q, R be the QR-decomposition of A. For every $x \in S$ it holds that:

$$\boldsymbol{f(A,x) = \|Ax\|^2 = \|QRx\|^2 = \|Rx\|^2 = f(R,x)}$$

$A = QR$ $\quad\quad$ $Q^T Q = I$

Hence, the rows of R are an exact coreset (yet not a subset of the data) for the Points-Hyperplane Query Space problem since:

$$\forall x \in S:$$

$$\boldsymbol{f(A,x) = \|Ax\|^2 = \|Rx\|^2 = f(R,x)}$$

$A \in \mathbb{R}^{n\times d}$ $\quad\quad$ $R \in \mathbb{R}^{d\times d}$

Exact Coreset - Example

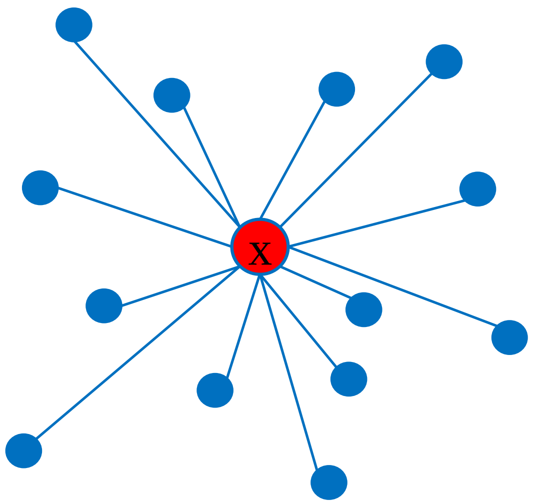
Problem: 1-mean

Input: The query space (P, w, Q, f) of 1-mean. We currently assume that $w(p) = 1$ for every $p \in P$.

Goal: Compute a pair (C, μ) such that

$$\forall \boldsymbol{x} \in \boldsymbol{Q}$$

$$\sum_{p \in P} \mathbf{1} \cdot \|\boldsymbol{p} - \boldsymbol{x}\|^2 = \sum_{c \in C} \boldsymbol{\mu}(\boldsymbol{c}) \cdot \|\boldsymbol{c} - \boldsymbol{x}\|^2$$



Exact Coreset - Example

Problem: 1-mean

Exact $1\text{-}mean$ **using 3 first moments:**

$$\sum_{\boldsymbol{p}\in\boldsymbol{P}} \|\boldsymbol{p}-\boldsymbol{x}\|^2 = \sum_{p\in P} \left(\|p\|^2 + \|x\|^2 - 2p^T x\right) = \sum_{p\in P} \|p\|^2 + \sum_{p\in P} \|x\|^2 - 2\sum_{p\in P} p^T x$$

$$= \sum_{p\in P} \|p\|^2 + \boldsymbol{n}\cdot\|\boldsymbol{x}\|^2 - \boldsymbol{2}\left(\sum_{p\in P} p^T\right)\boldsymbol{x}$$

The statistics that define the set P

Solution #1:
Store the three statistics in memory.

However, they do not satisfy our definition of Exact Coreset!

Exact Coreset - Example

Problems with solution #1:

- If the input data is sparse, the vector $\sum_{p \in P} p^T$ might not be sparse!
- The vector $\sum_{p \in P} p^T$ is not part of the input data. We prefer our representatives to be a subset of the input data!

Solution #2:

Try to find a an exact coreset (subset) C of the data and a weights $\omega: C \to R$ such that:

$$\sum_{p \in P} \|p\|^2 = \sum_{c \in C} \omega(c)\|c\|^2$$

$$|\mathrm{P}| = \boldsymbol{n} = \sum_{c \in C} \omega(c)$$

$$\sum_{p \in P} p^T = \sum_{c \in C} \omega(c) c^T$$

1-mean queries

Solution #2:

1) Build new vectors in R^{d+2}:

$$\boldsymbol{p}_i' = \begin{pmatrix} \boldsymbol{p}_i \\ \|\boldsymbol{p}_i\|^2 \\ \mathbf{1} \end{pmatrix}$$

2) Find a *Linear Combination* of the mean of P. This combination is a subset $C \subseteq P$ of the vectors of size $|C| = d + 2$ and a set μ of $d + 2$ weights.
The set C satisfies the three properties needed.

Problem with solution #2:

The weights are not bounded (might be negative and huge → numerical problems).

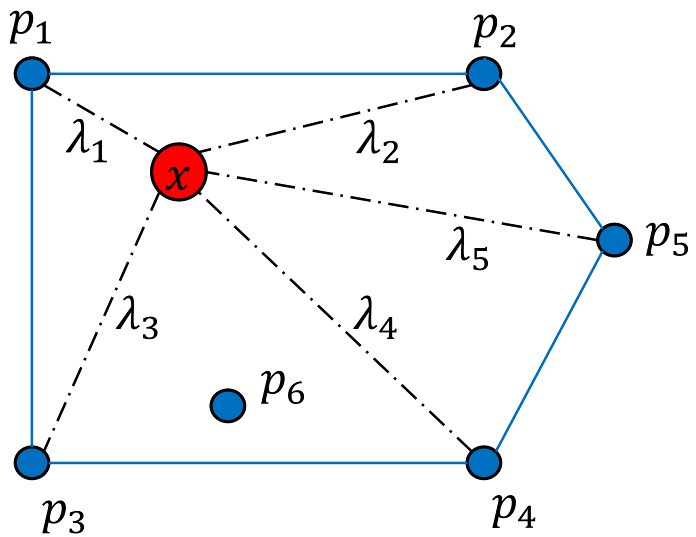
Preliminaries - Convex combination

A *convex combination* is a linear combination of points where all coefficients are non-negative and sum to 1.

A *convex region* is a region where, for every pair of points within the region, every point on the straight line segment that joins the pair of points is also within the region.

A *convex hull* of a set P is is the smallest convex set that contains P.

Every point x in a convex hull of a set of points P can be written as a convex combination of a finite number of points in P.



$$x = \sum_{i=1}^{5} \lambda_i p_i ,$$

$$\lambda_i \geq 0, \sum_{i=1}^{5} \lambda_i = 1$$

1-mean queries

Solution #3:

1) Build new vectors in R^{d+2}:

$$\boldsymbol{p}_i' = \begin{pmatrix} \boldsymbol{p}_i \\ \|\boldsymbol{p}_i\|^2 \\ \mathbf{1} \end{pmatrix}$$

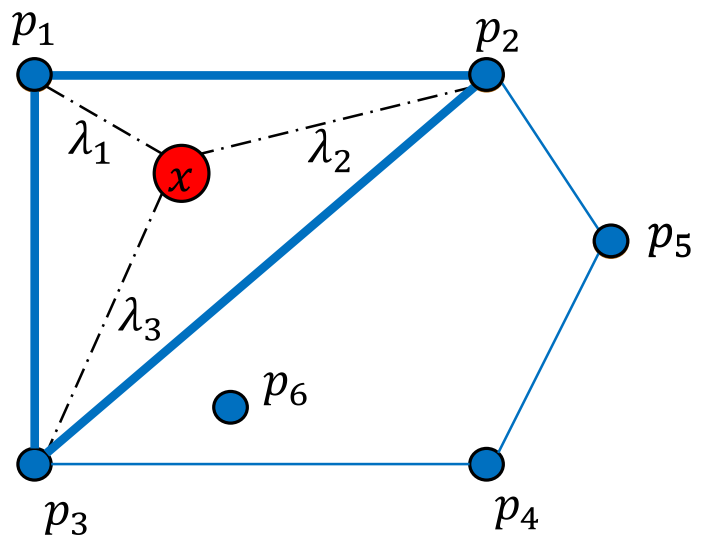
2) Find a $Convex\ Combination$ of the mean of P using $Caratheodory's\ theorem$.
This combination is a subset $C \subseteq P$ of the vectors of size $|C| = d + 3$ and a set μ of $d + 3$ positive weights that sum to one.
The set C satisfies the three properties needed.

Caratheodory's theorem

"If a point $x \in R^d$ lies in the convex hull of a set P, there is a subset P' of P consisting of $d + 1$ or fewer points such that x lies in the convex hull of P'."

$$x = \sum_{i=1}^{3} \lambda_i p_i ,$$

$$\lambda_i \geq 0, \sum_{i=1}^{3} \lambda_i = 1$$



Caratheodory's theorem - intuition

Convex combination: λ_i. All are positive.

Linear combination: μ_i. One of them is negative.

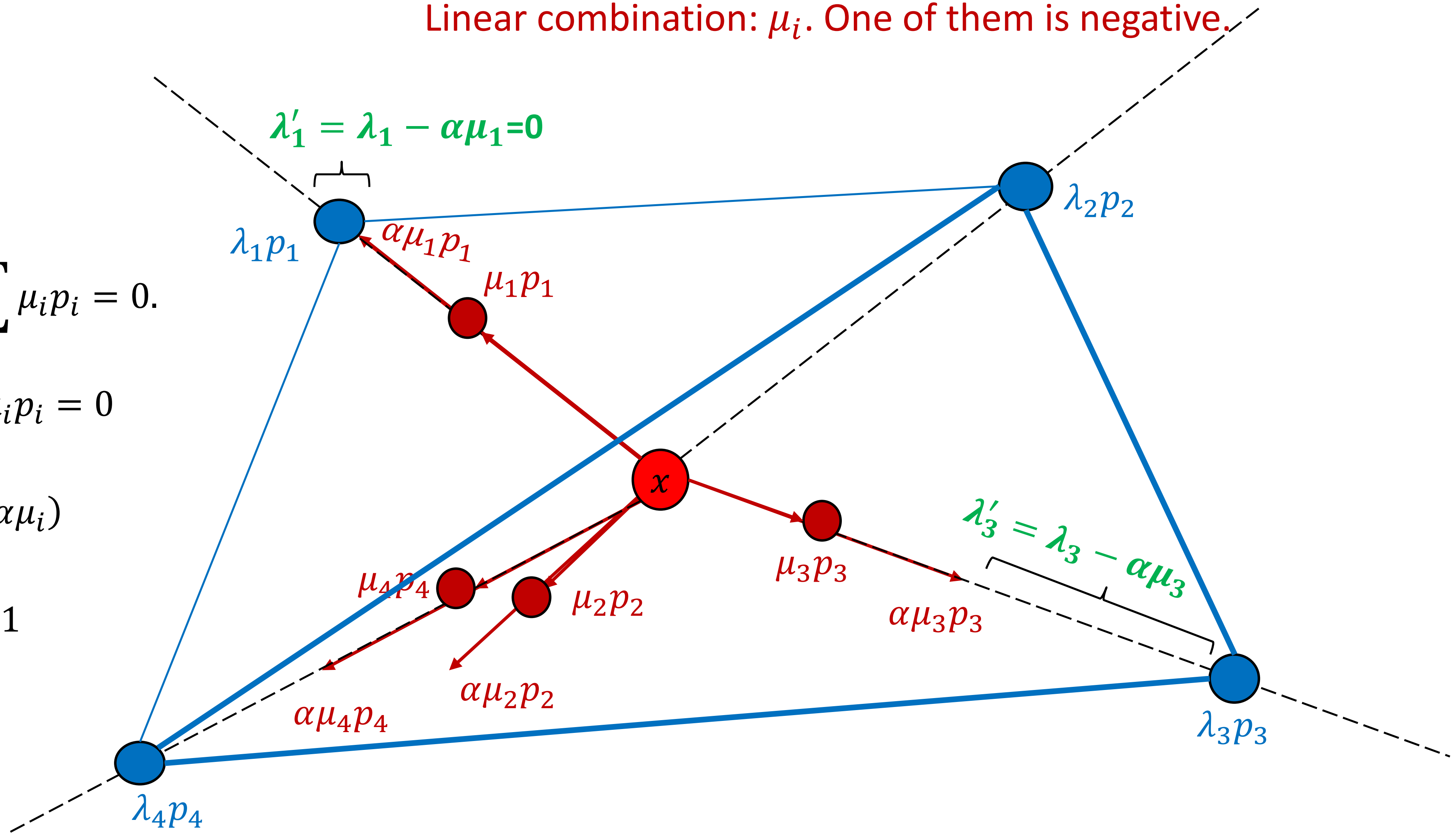
Assume that x is the origin.

$$\sum \lambda_i = 1, \sum \mu_i = 0, \sum \mu_i p_i = 0.$$

$$\rightarrow \sum \alpha \mu_i p_i = \alpha \sum \mu_i p_i = 0$$

$$\rightarrow \sum \lambda_i{}' = \sum (\lambda_i - \alpha \mu_i)$$

$$= \sum \lambda_i - \alpha \sum \mu_i = 1$$



1-mean queries

Solution #3:

Using *Caratheodory's Theorem* we can represent the vector

$$\frac{\sum_{i=1}^{n} p_i'}{n} = \frac{\sum_{i=1}^{n} \begin{pmatrix} p_i \\ \|p_i\|^2 \\ 1 \end{pmatrix}}{n} = \frac{\begin{pmatrix} \sum_{i=1}^{n} p_i \\ \sum_{i=1}^{n} \|p_i\|^2 \\ n \end{pmatrix}}{n}$$

by a set of $(d+3)$ input points and $(d+3)$ weights.

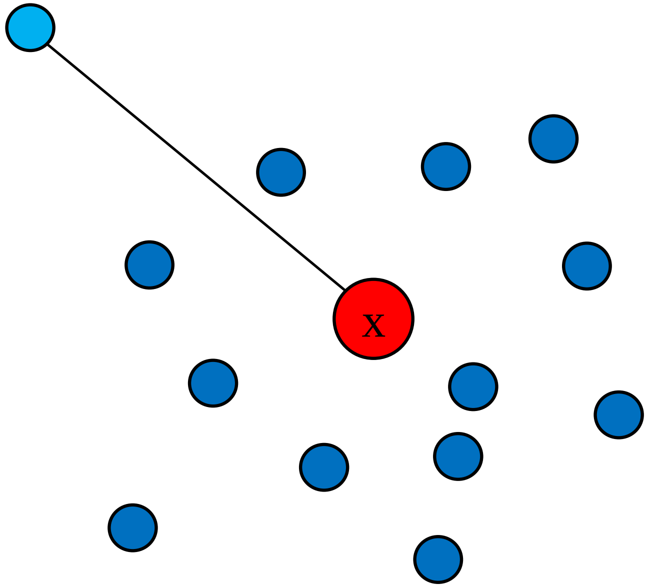
Saved in memory

$\lambda_1 *$	0	0	0	**1**	0	0	0	0	0	0	0	0	.	.	.	0	**1**	p'_1
$\lambda_2 *$	0	0	0	**1**	0	0	0	0	0	0	0	0	.	.	.	0	**1**	p'_2
⋮																		
$\lambda_{d+3} *$	0	0	0	**1**	0	0	0	0	0	0	0	0	.	.	.	0	**1**	p'_{d+3}

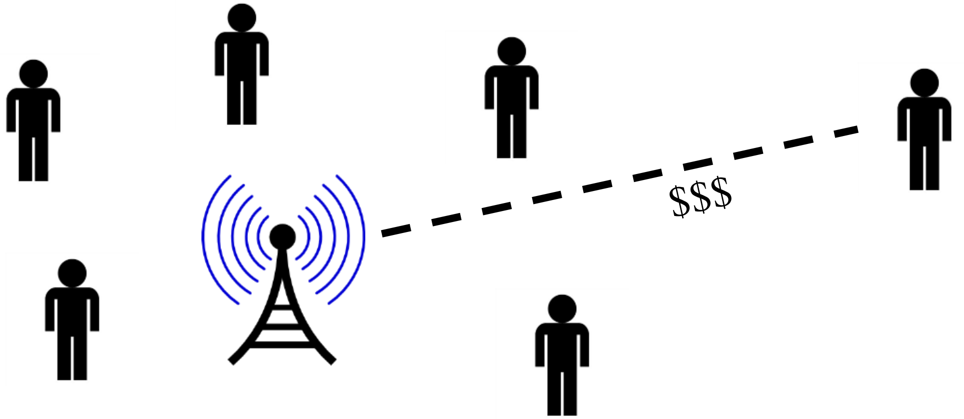
1-center / minimum enclosing ball

- Given a set of n points P in R^d, find the point $q \in R^d$ that minimizes:

$$\boldsymbol{far(P,q) = \max_{p \in P} \|p - q\|}$$



Motivation:
Where should we place an antenna if the price paid is the antenna's distance to the farthest customer?



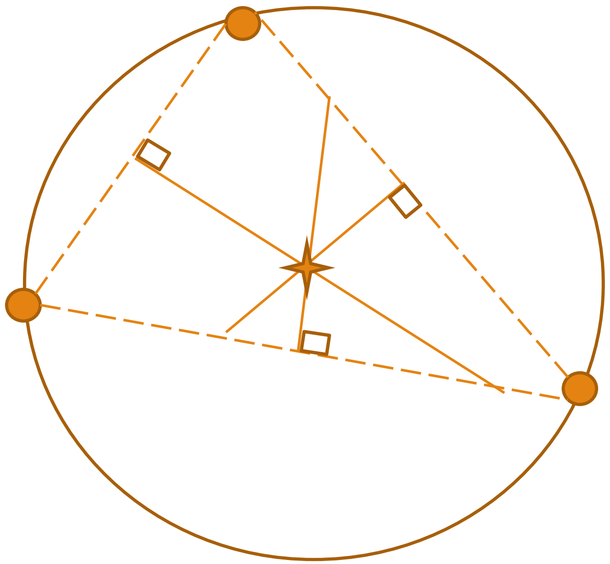
Minimum enclosing ball

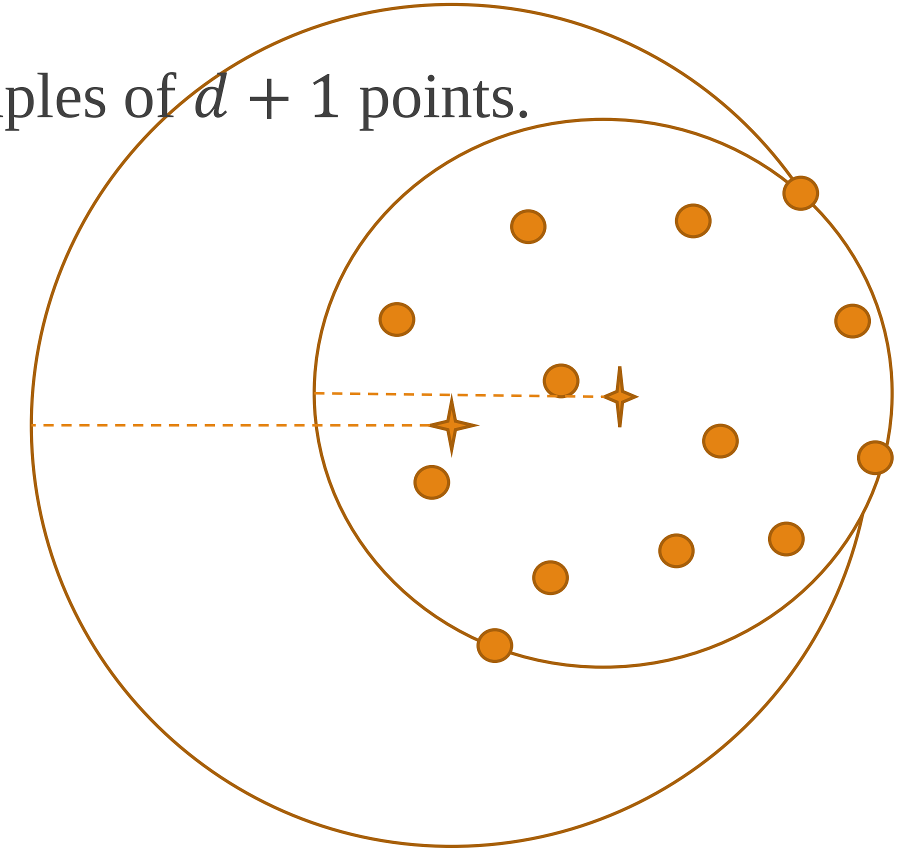
<u>Optimal solution in R^d:</u>

Claim: A sphere in R^d is determined by $d+1$.

Algorithm: Exhaustive search over all $\binom{n}{d+1}$ tuples of $d+1$ points.

Running time: $n^{O(d)}$.





Minimum enclosing ball - heuristic

Hough transform:

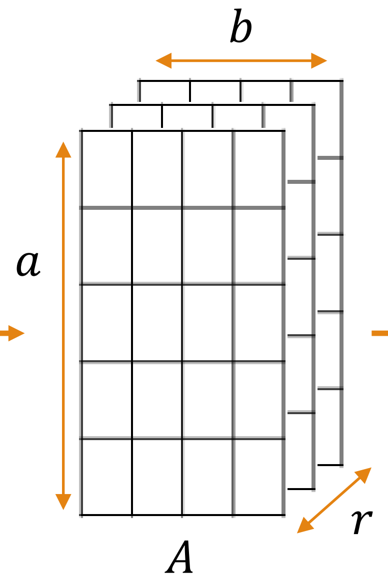
- A heuristic for finding a circle that best fits the data.
- Divides the circle parametric space into small fixed-size cells (grid) with no optimality guarantee. (Assumes circle radius is in range $[r_1, r_2]$).

$$Let\ circ(x, y, r) = \{(a, b) \mid \|a - x, b - y\| = r\}.$$

$n\ points$ → $for\ each\ point\ (x, y)$ →

$for\ every\ radius\ r \in [r_1, r_2]$
$\quad for\ every\ point\ (a, b) \in circ(x, y, r)$
$\quad\quad A[a, b, r] + +;$

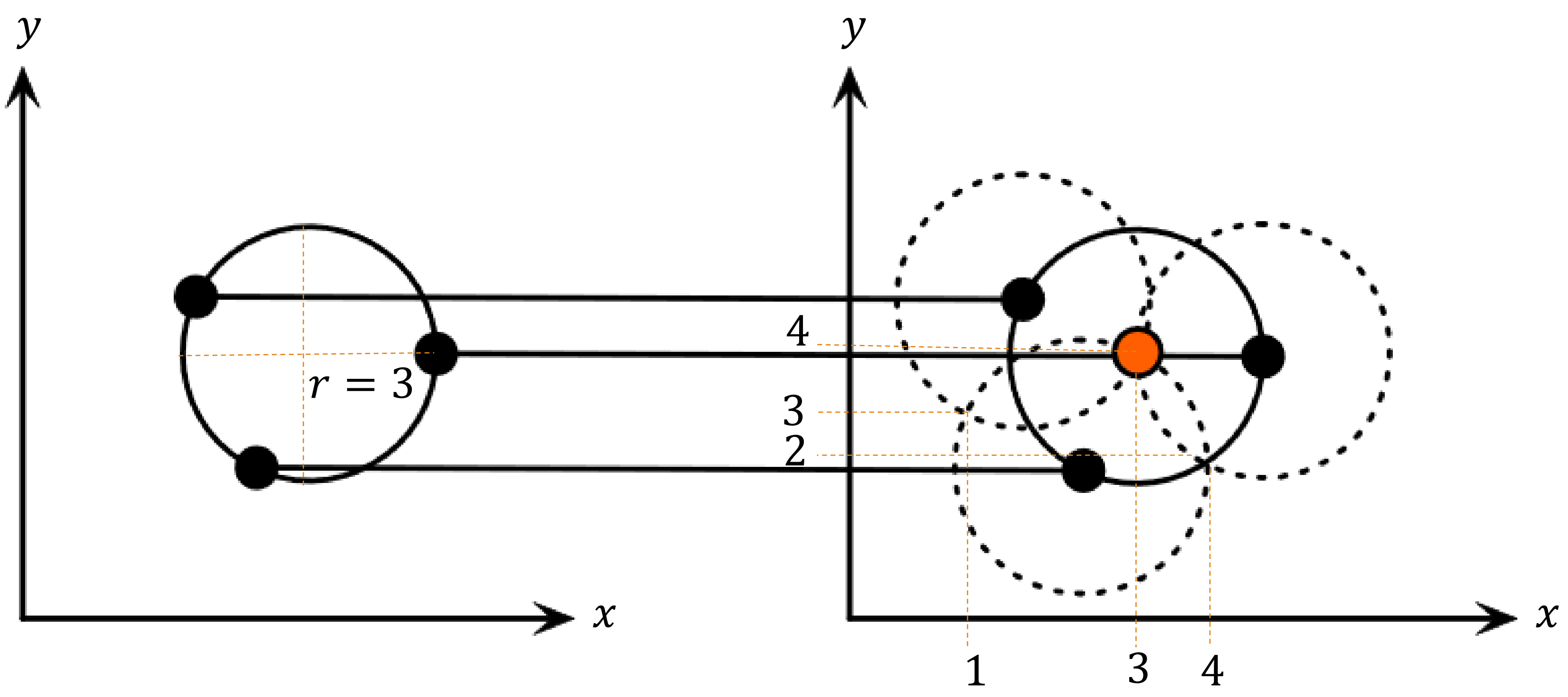
Assume every data point is a circle center, and "vote" for each point on that circle.



→ $Find\ global\ maxima\ of\ A$

The global maxima are the "best fitting" circles (the circles with maximal number of votes)

Hough transform - example



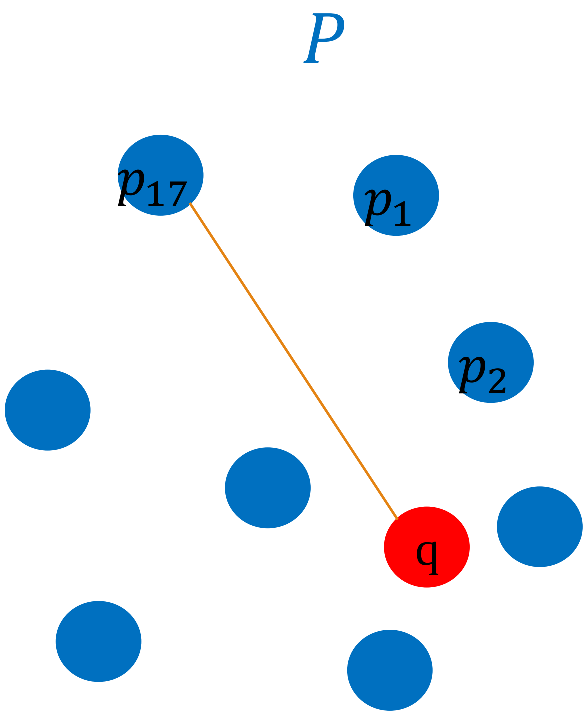
	1	2	3	4
1				0
2				
3	2	0		**3**
4		2		

$A_{r=3}$

Query Space

Problem: One center

- $P = \{p_1, \cdots, p_n\} \subseteq \mathbb{R}^d$.
- $Q = \mathbb{R}^d$ (every possible point in $\mathbb{R}^d$).
- $\omega(P) = 1$ for every $p \in P$.
- $far(P, q) = \max_{p \in P} \|p - q\|$ for every $q \in Q$.



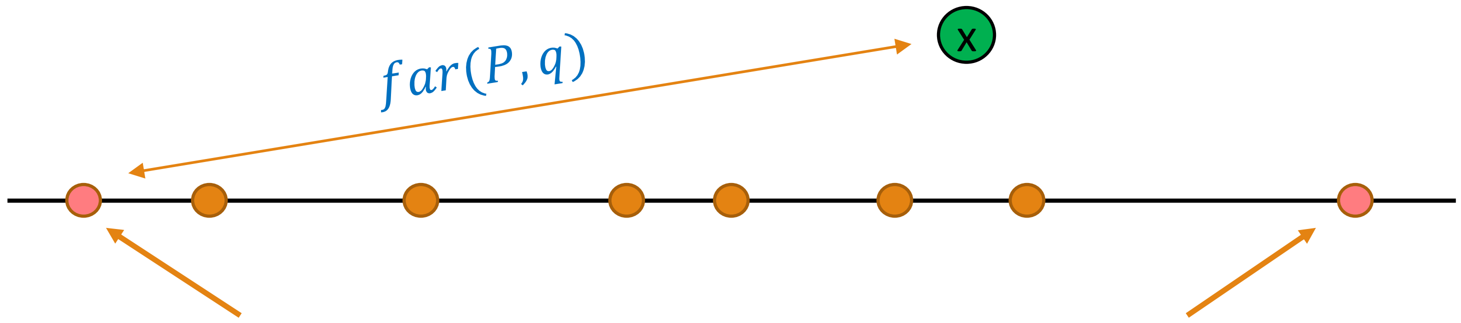
The tuple (P, ω, Q, far) is our query space.

1-center

Input: The query space (P, w, Q, far) of one center.

Special case:

Exact coreset for $1\text{-}center$ queries when $P \subset R$ and $q \in R^d$:



The farthest point from every query $q \in R^d$ is one of the edge points!

ε-Coresets

Let A be a set of elements. Let (P, ω, Q, f) be a query space where $P \subseteq A$.

An ε-coreset is a set (C, μ), where $C \subseteq A, \mu: C \to [0, \infty)$, such that for every $q \in Q$ we have that:

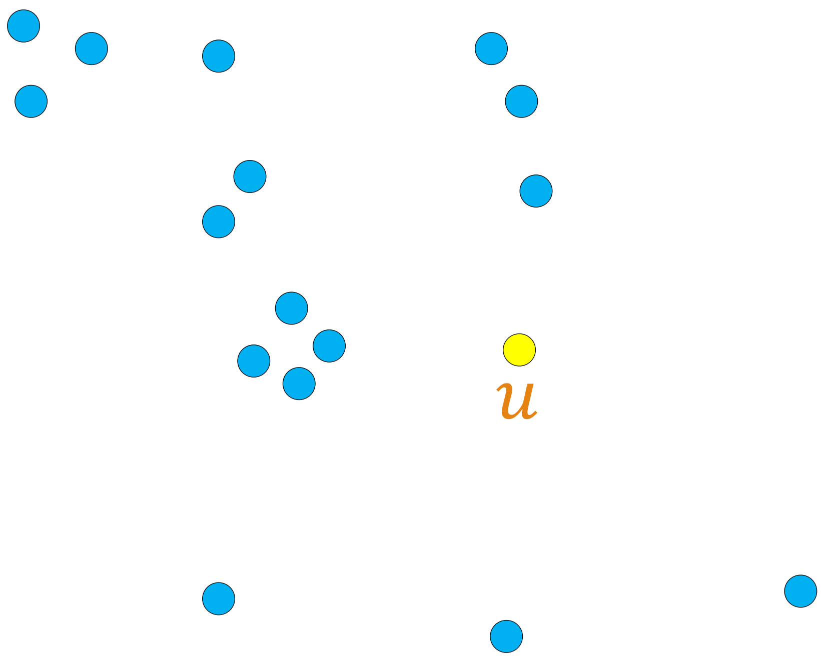
$$(1-\varepsilon) \cdot \sum_{p \in P} \omega(p) \cdot f(p,q) \leq \sum_{c \in C} \mu(c) \cdot f(c,q) \leq (1+\varepsilon) \cdot \sum_{p \in P} \omega(p) \cdot f(p,q)$$

$$\Leftrightarrow$$

$$\left| \sum_{p \in P} \omega(p) \cdot f(p,q) - \sum_{c \in C} \mu(c) \cdot f(c,q) \right| \leq \varepsilon \cdot \sum_{p \in P} \omega(p) \cdot f(p,q)$$

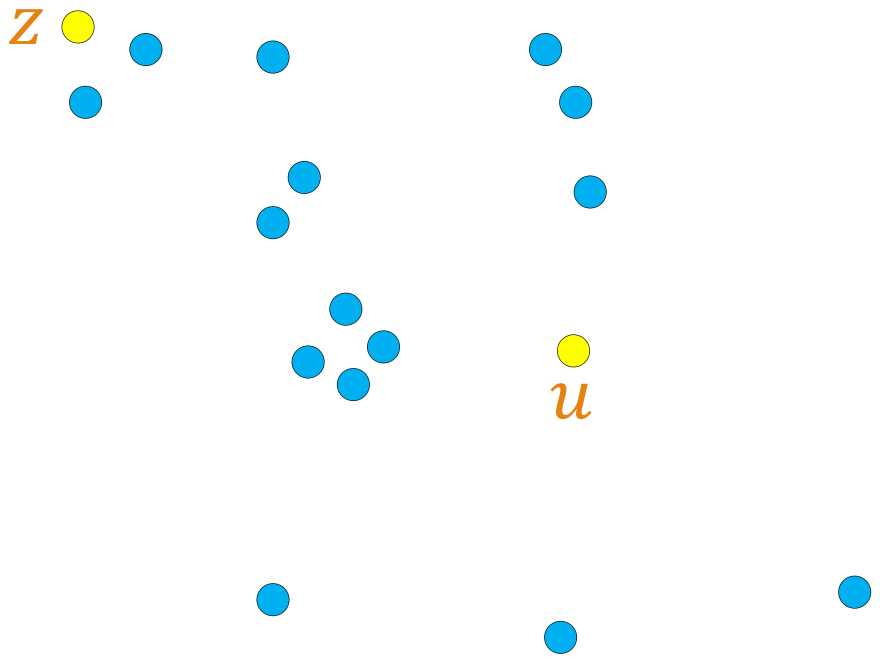
ε-Coreset for 1-Center / Enclosing Balls

1) Choose an arbitrary point $u \in P$



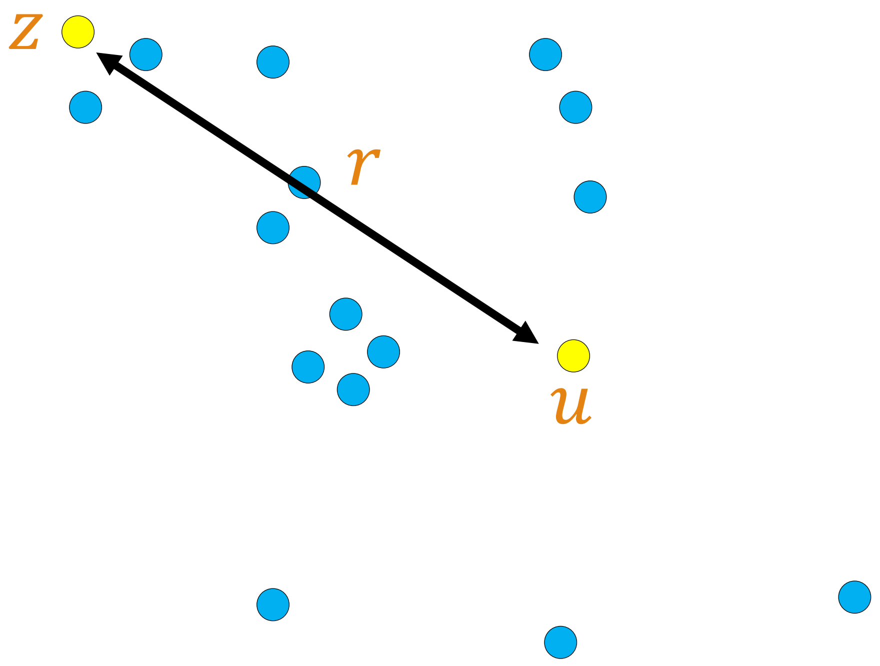
ε-Coreset for 1-Center / Enclosing Balls

2) Find the farthest point $z \in P$ from u



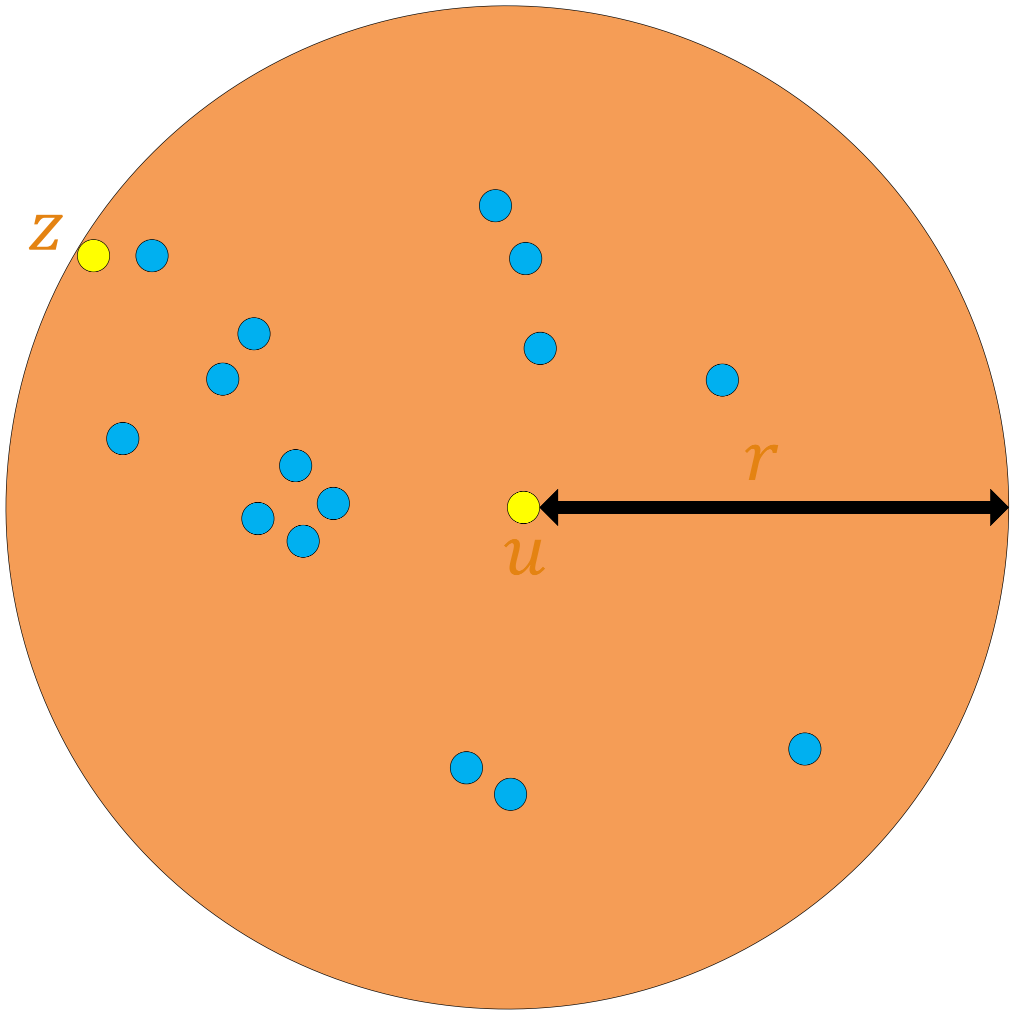
ε-Coreset for 1-Center / Enclosing Balls

$r \coloneqq dist(u, z)$



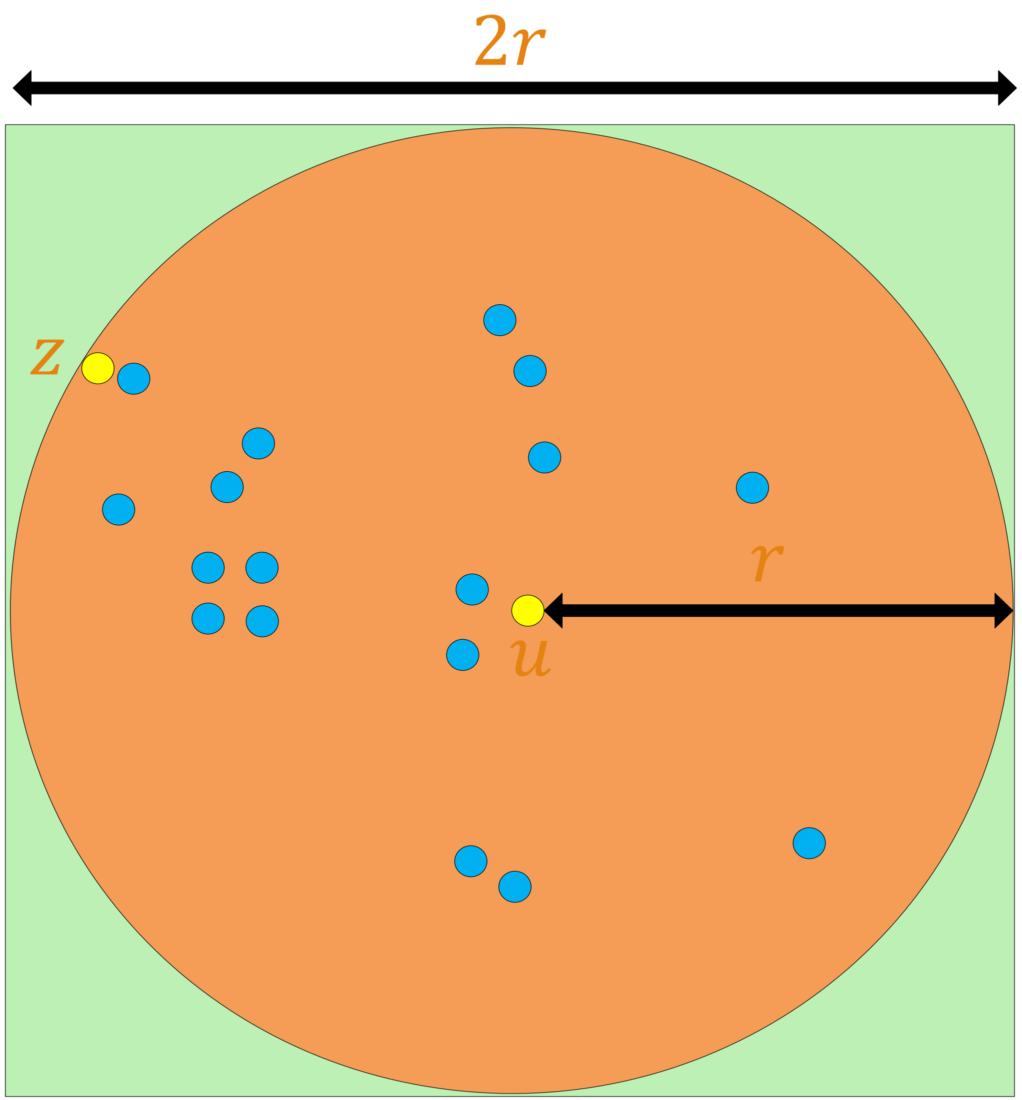
ε-Coreset for 1-Center / Enclosing Balls

$r \coloneqq dist(u, z)$



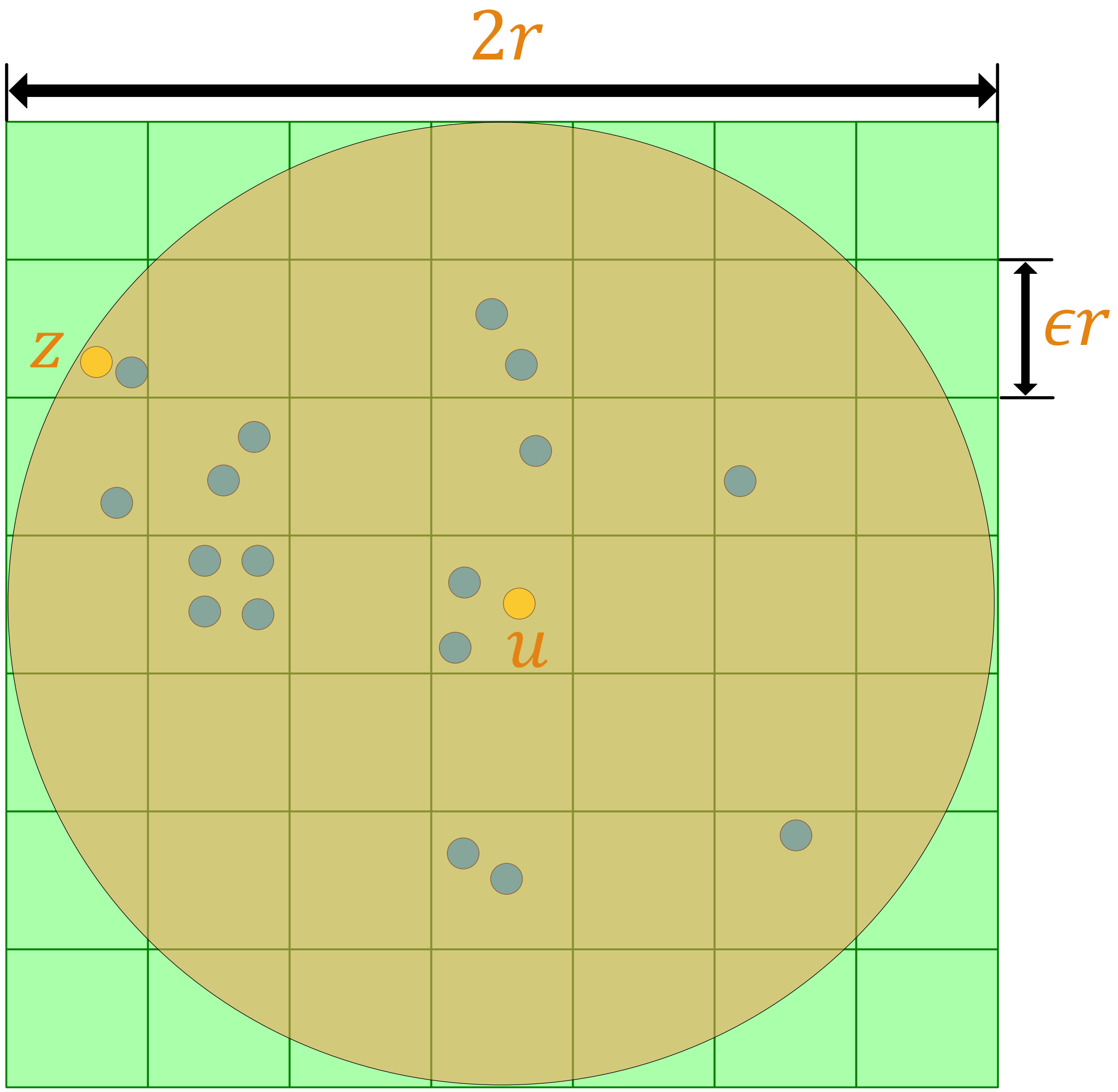
ε-Coreset for 1-Center / Enclosing Balls

$r \coloneqq dist(u, z)$



ε-Coreset for 1-Center / Enclosing Balls

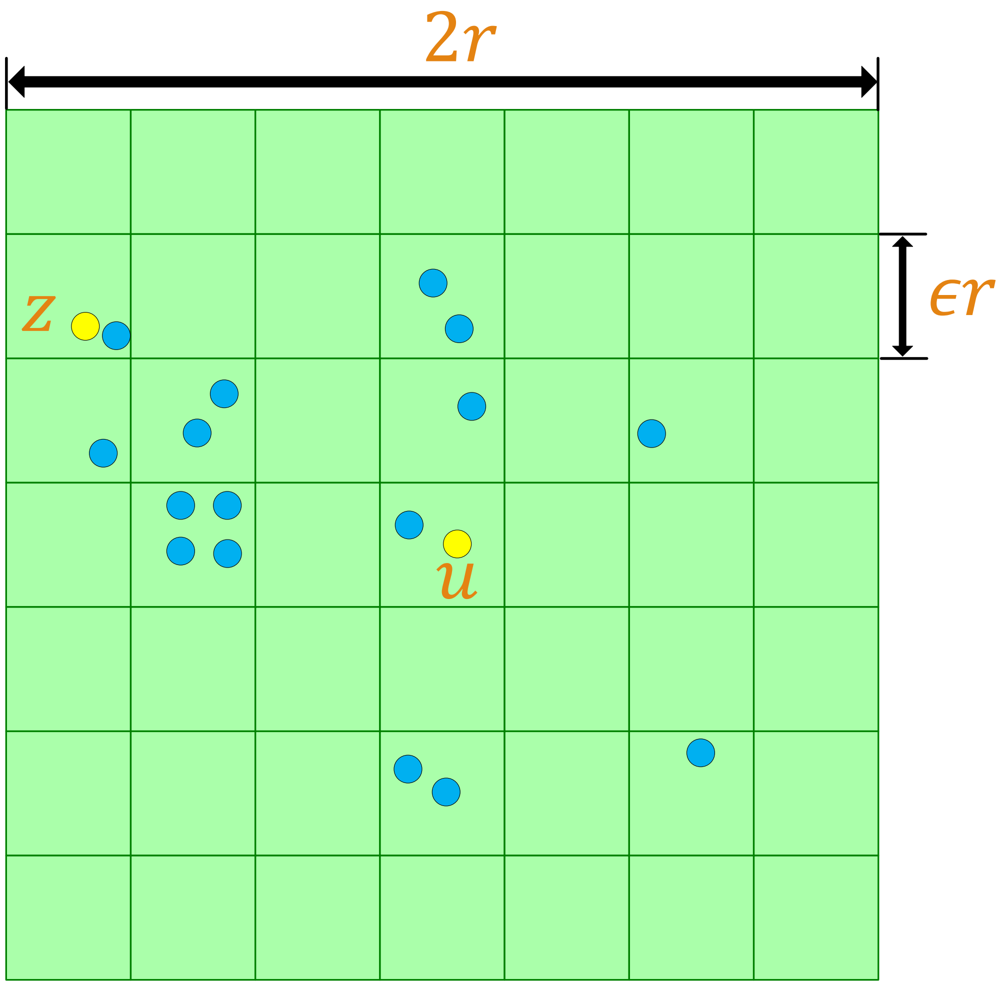
$r \coloneqq dist(u, z)$



ε-Coreset for 1-Center / Enclosing Balls

3) Construct a grid of $\frac{2}{\epsilon^2}$ cells of size $\epsilon r \times \epsilon r$, centered at u

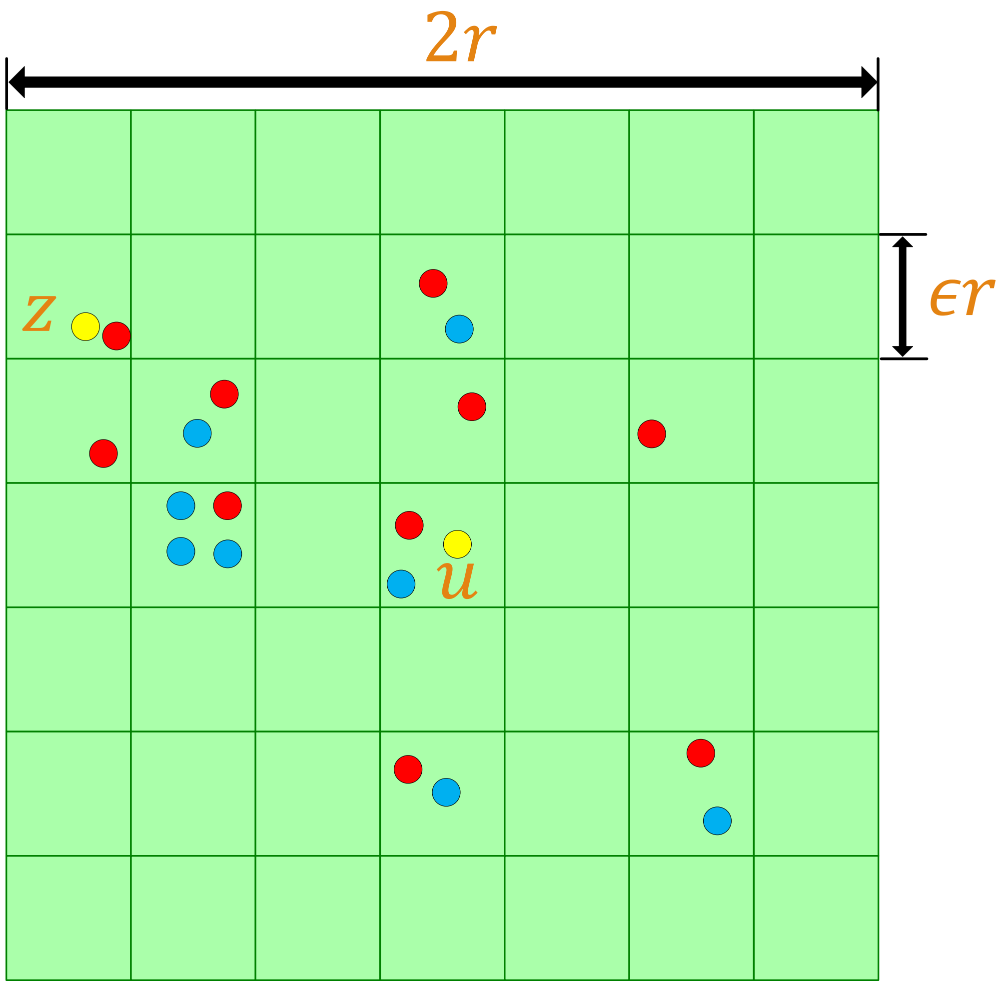
$$r \coloneqq dist(u, z)$$



ε-Coreset for 1-Center / Enclosing Balls

4) Pick a representative point from each non-empty cell

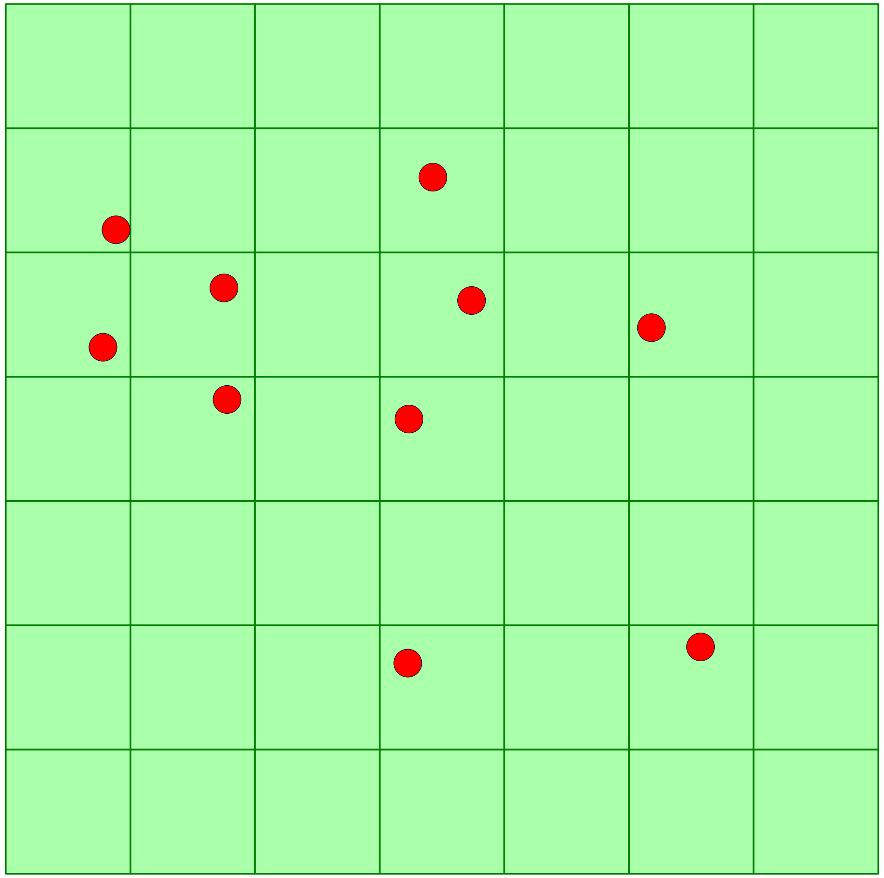
$r \coloneqq dist(u, z)$



ε-Coreset for 1-Center / Enclosing Balls

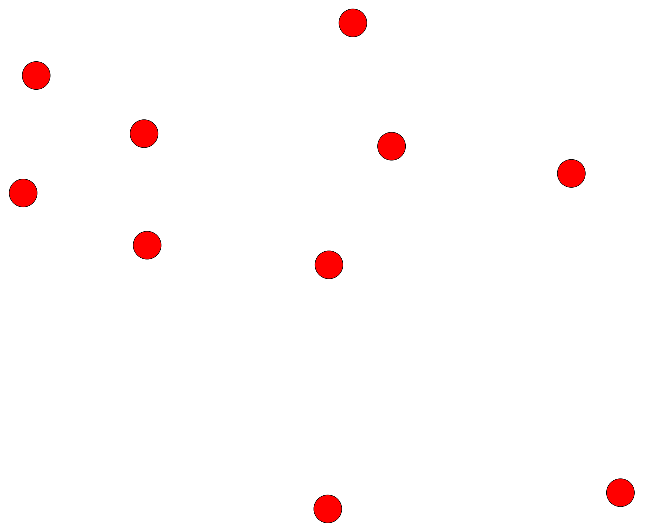
5) C $\coloneqq$ the set of

the $O\left(\frac{1}{\epsilon^2}\right)$ representatives



ε-Coreset for 1-Center / Enclosing Balls

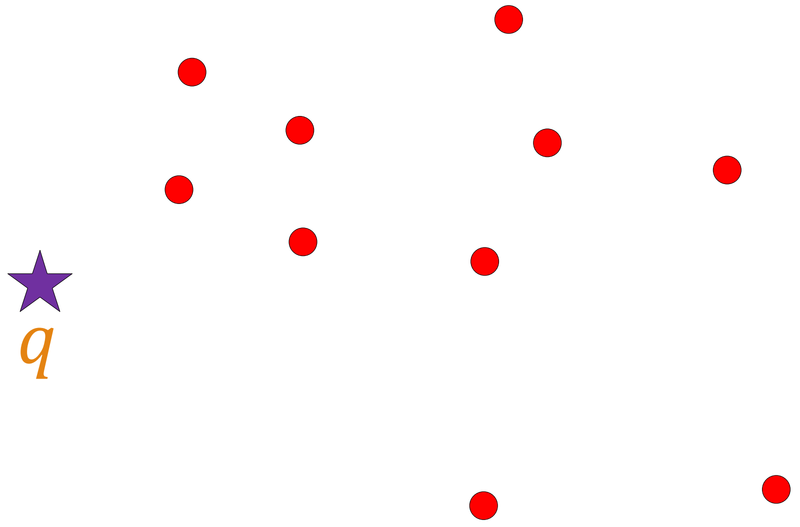
5) Return C



ε-Coreset for 1-Center / Enclosing Balls

Proof of Correctness

$q \coloneqq$ an arbitrary query point

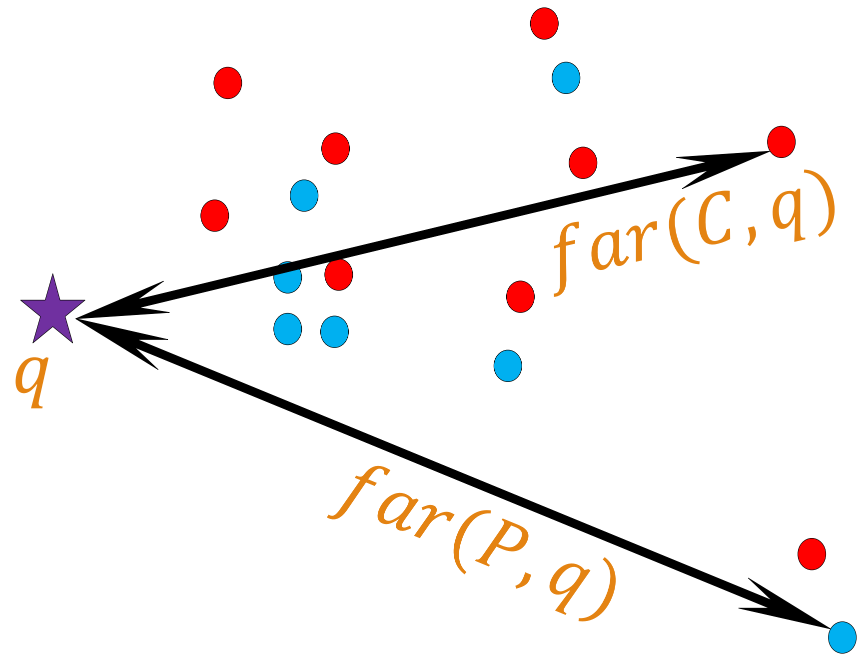


ε-Coreset for 1-Center / Enclosing Balls

Proof of Correctness

$$far(P,q) = \max_{p \in P} dist(p,q)$$

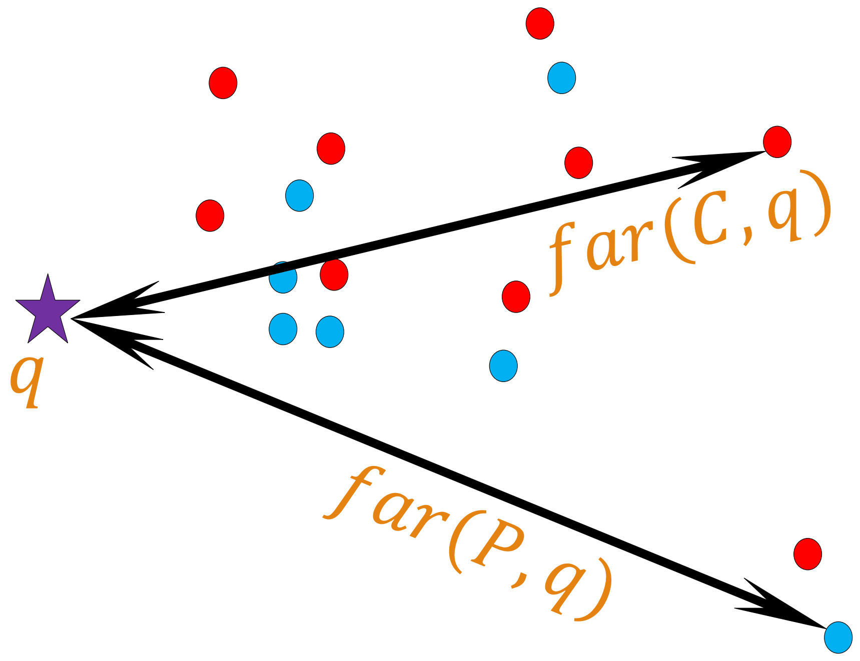
$$far(C,q) = \max_{c \in C} dist(c,q)$$



ε-Coreset for 1-Center / Enclosing Balls

Proof of Correctness

$C \subseteq P \;\rightarrow\; far(C,q) \leq far(P,q)$



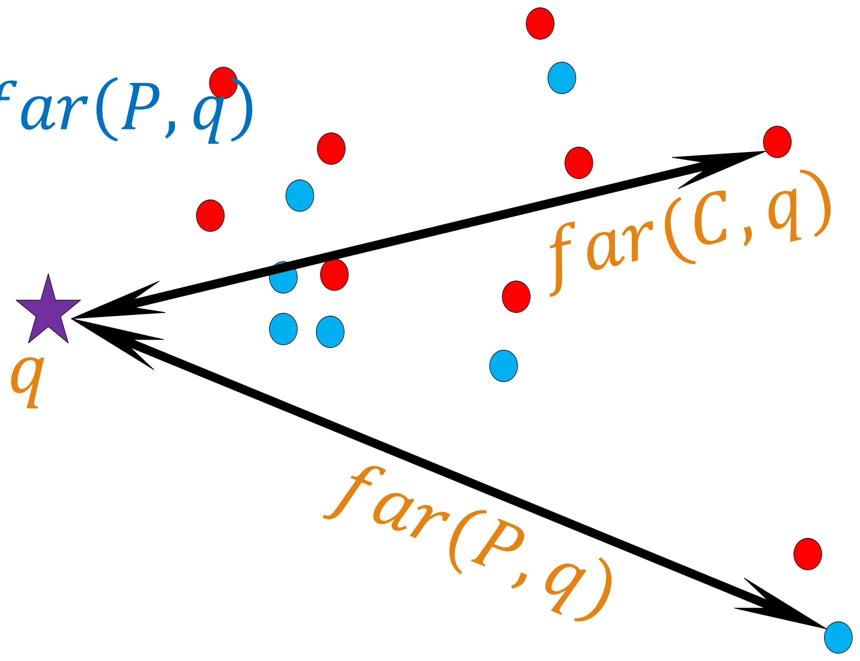
ε-Coreset for 1-Center / Enclosing Balls

Proof of Correctness

$$C \subseteq P \;\rightarrow\; far(C,q) \leq far(P,q)$$

Need to proof :

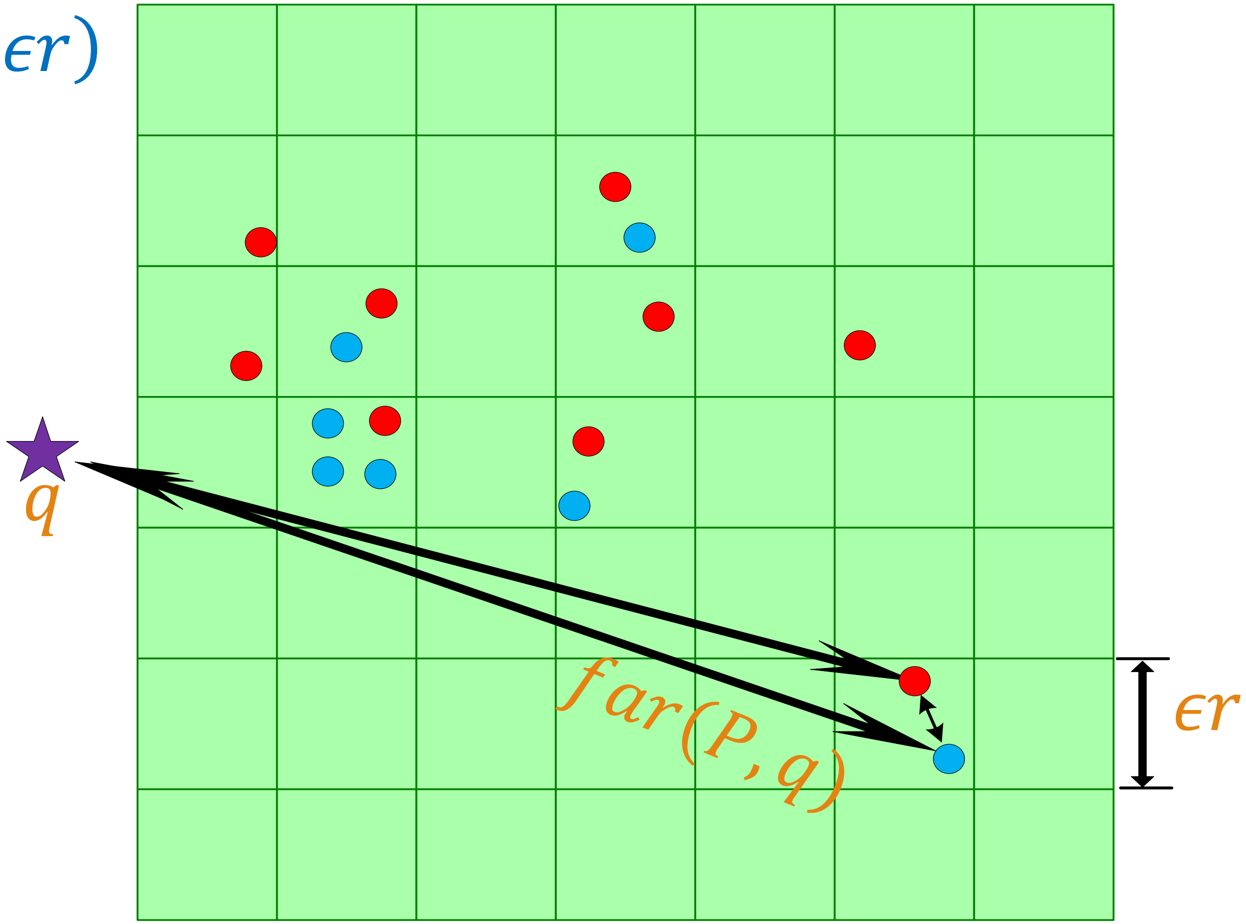
$$far(P,q) - far(C,q) \leq O(\epsilon) far(P,q)$$



ε-Coreset for 1-Center / Enclosing Balls

Proof of Correctness

$$far(P,q) \leq far(C,q) + O(\epsilon r)$$



ε-Coreset for 1-Center / Enclosing Balls

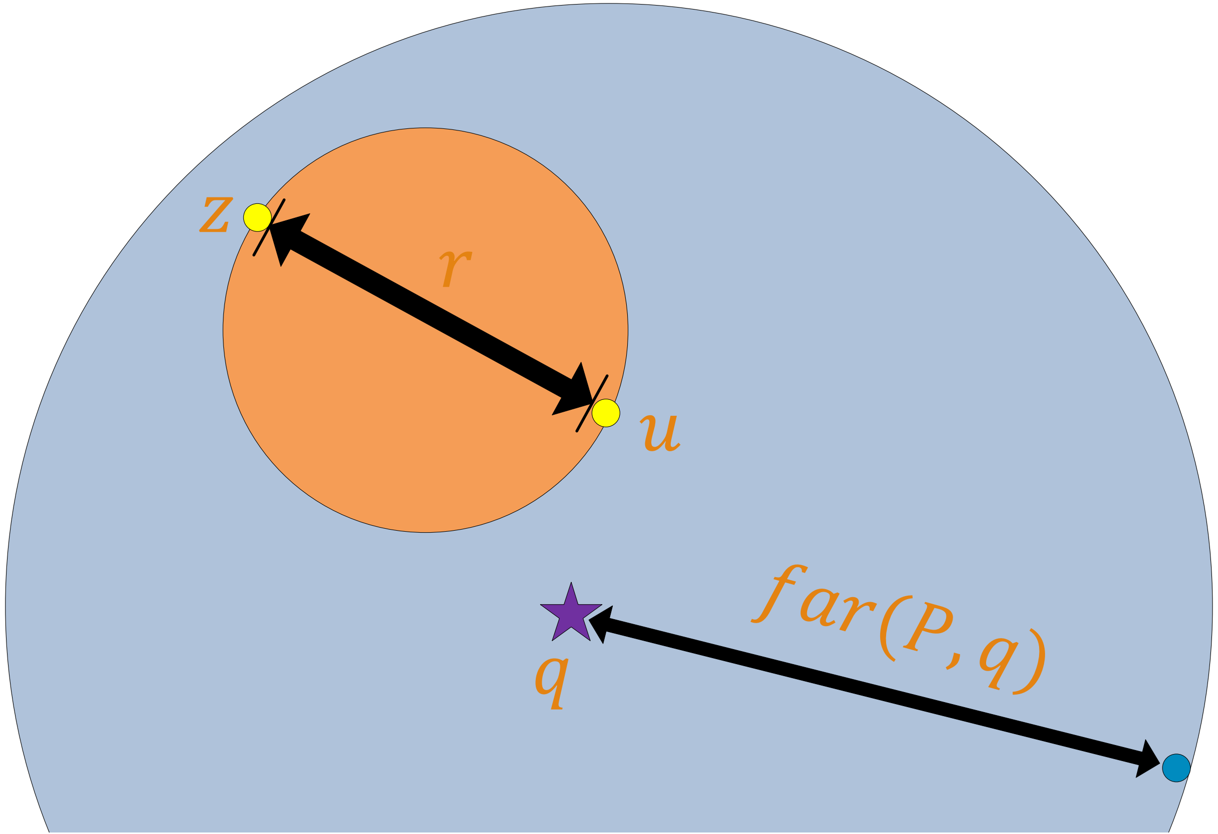
Proof of Correctness

Main observation:

Every ball that covers u and z, has a diameter of at least r.

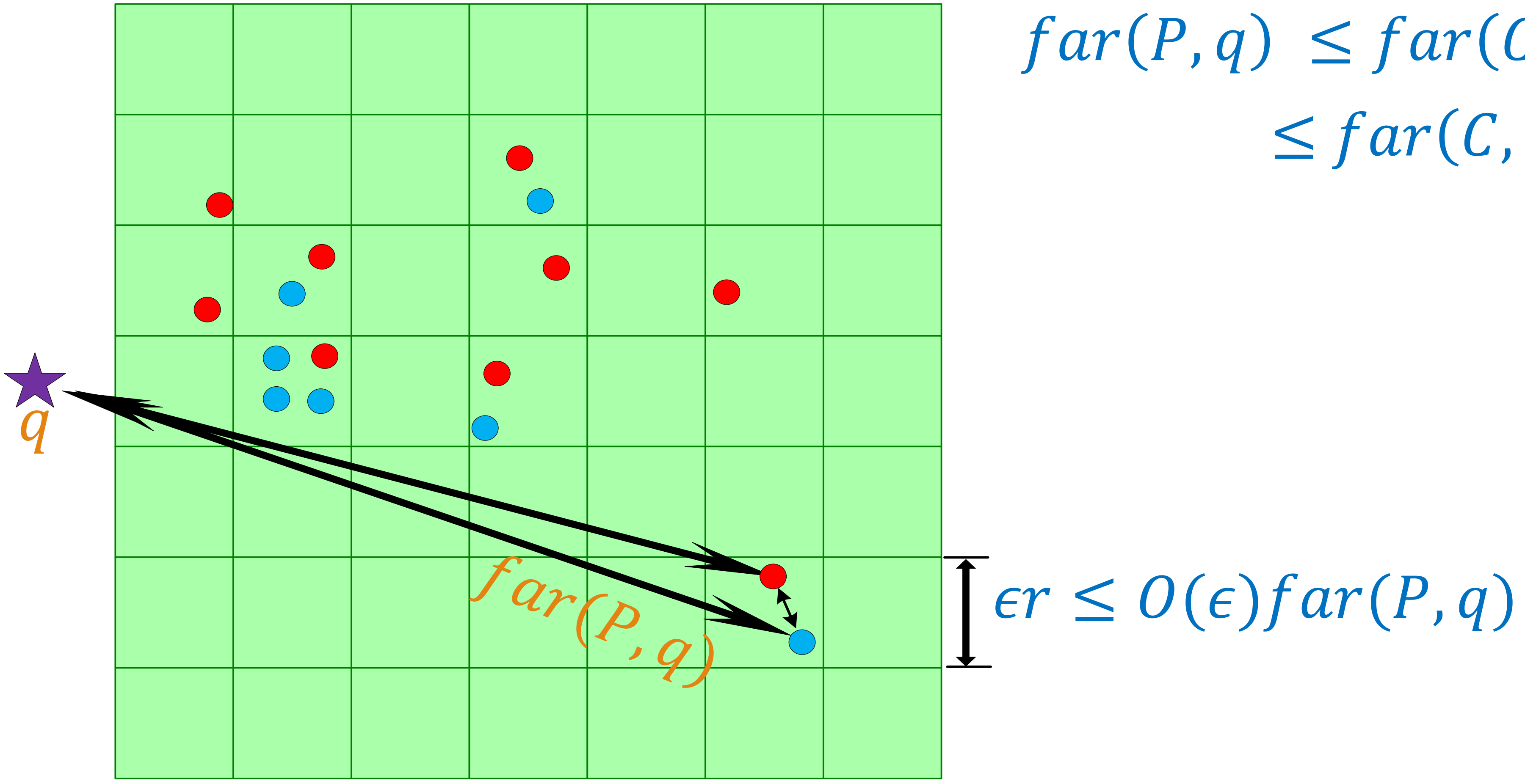
$$r \leq 2far(P,q)$$

$$O(\epsilon r) \leq O(\epsilon)far(P,q)$$



ε-Coreset for 1-Center / Enclosing Balls

Proof of Correctness

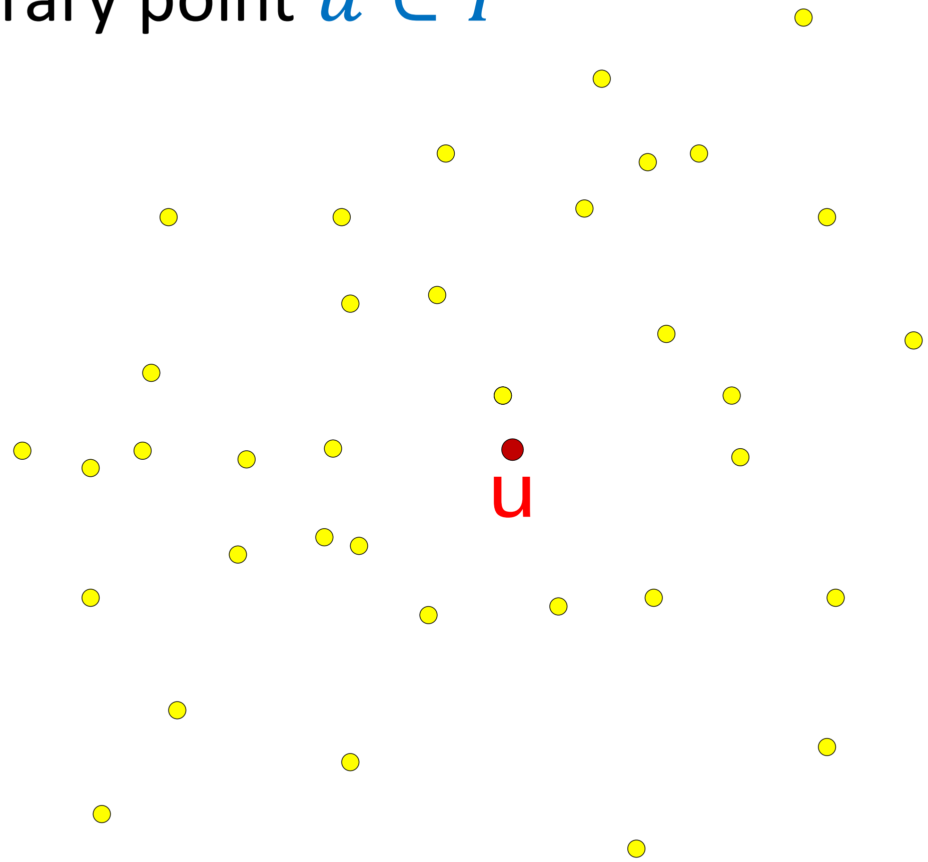


$$far(P,q) \le far(C,q) + O(\epsilon r)$$
$$\le far(C,q) + O(\epsilon) far(P,q)$$

ε-Coreset for 1-Center / Enclosing Balls

Smaller Coreset

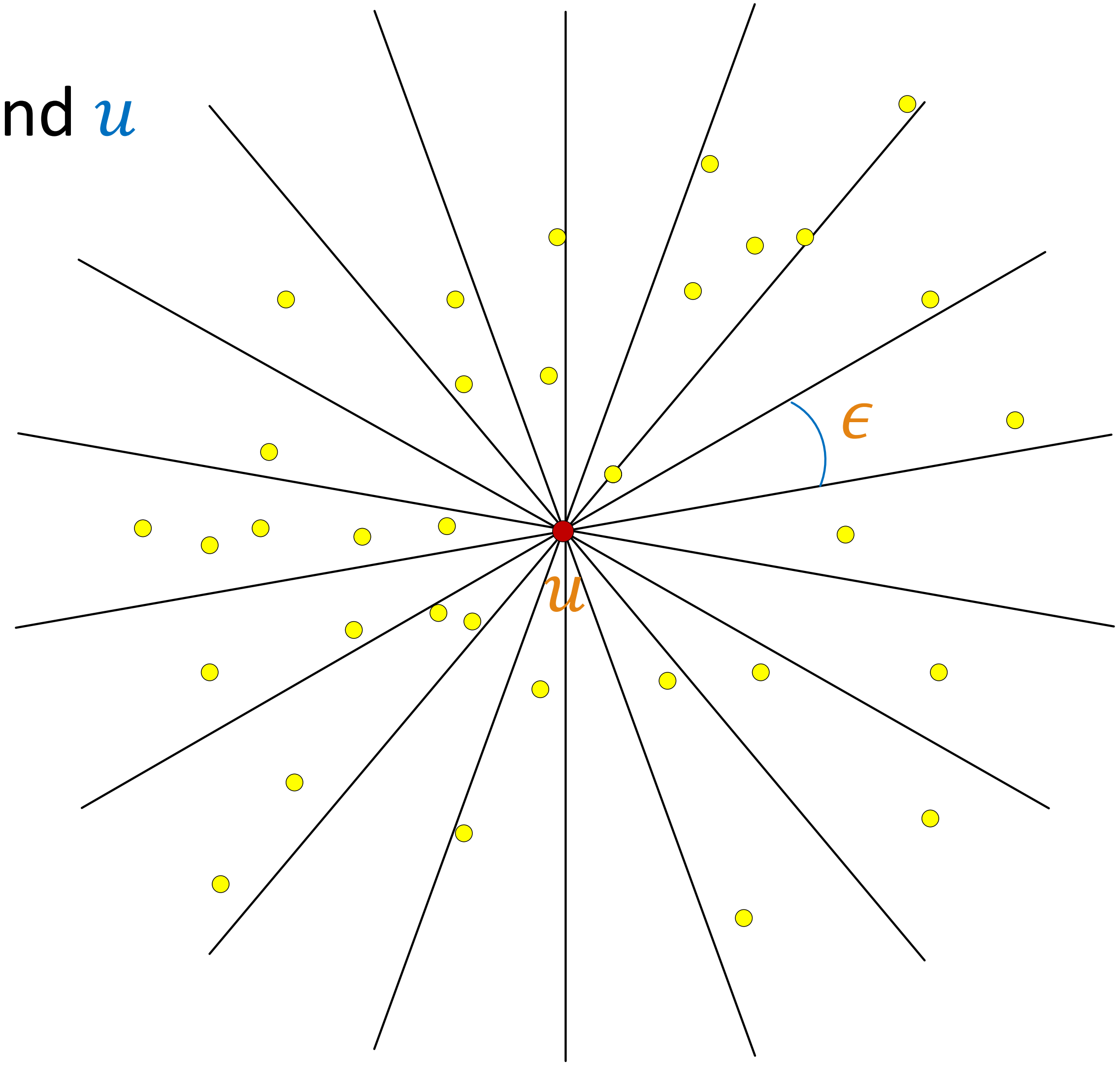
1) Choose an arbitrary point $u \in P$



ε-Coreset for 1-Center / Enclosing Balls

Smaller Coreset

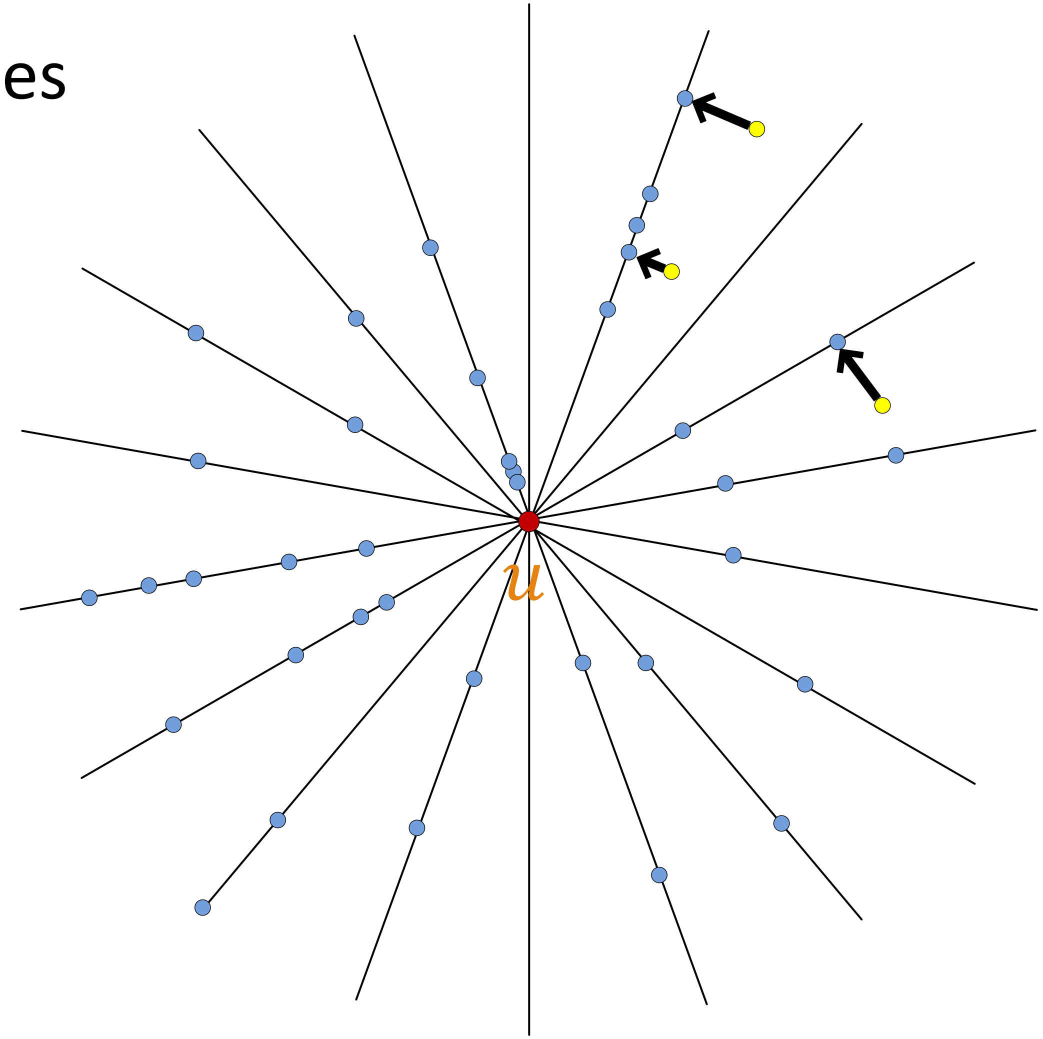
2) Draw a "star" of $\left(\frac{2\pi}{\epsilon}\right)$ lines around u



ε-Coreset for 1-Center / Enclosing Balls

Smaller Coreset

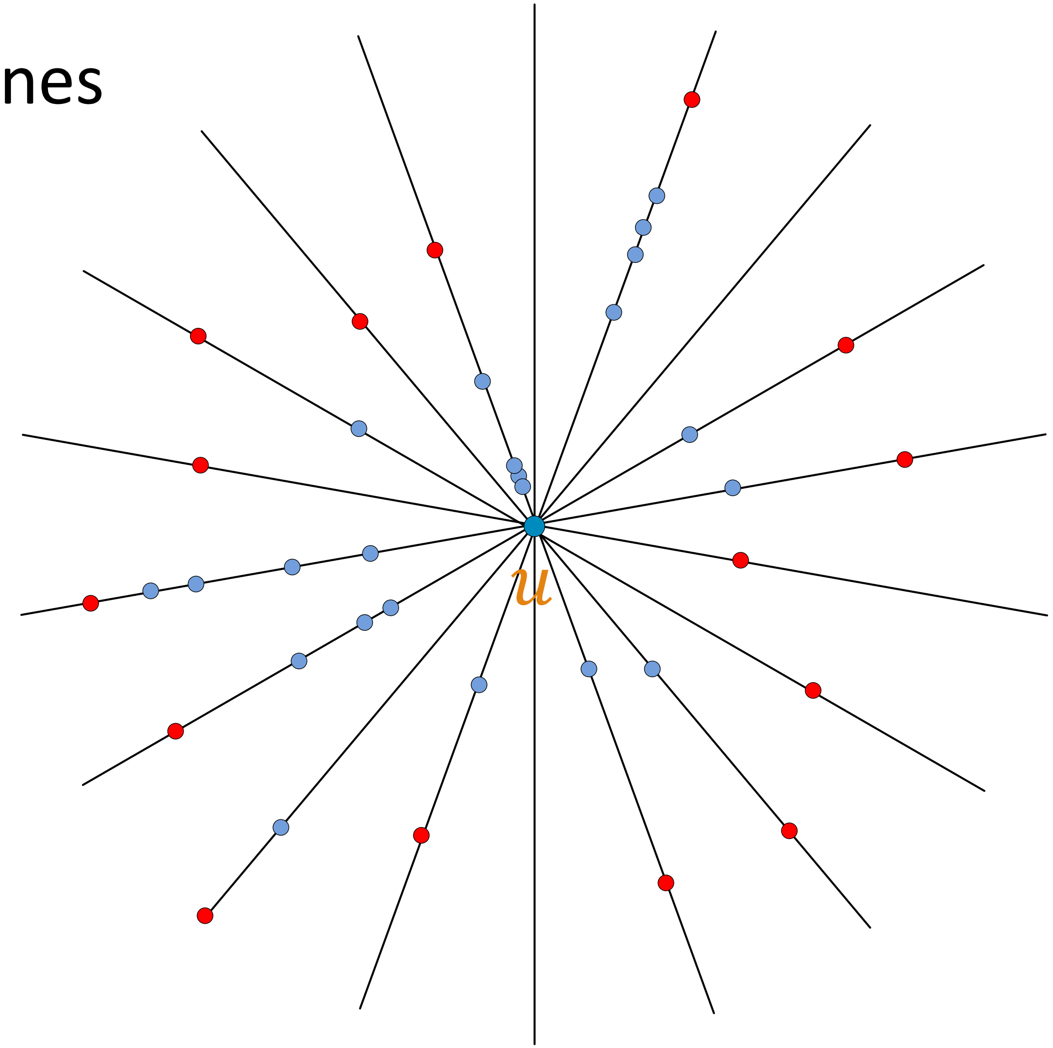
3) P' := Projection of P onto the lines



ε-Coreset for 1-Center / Enclosing Balls

Smaller Coreset

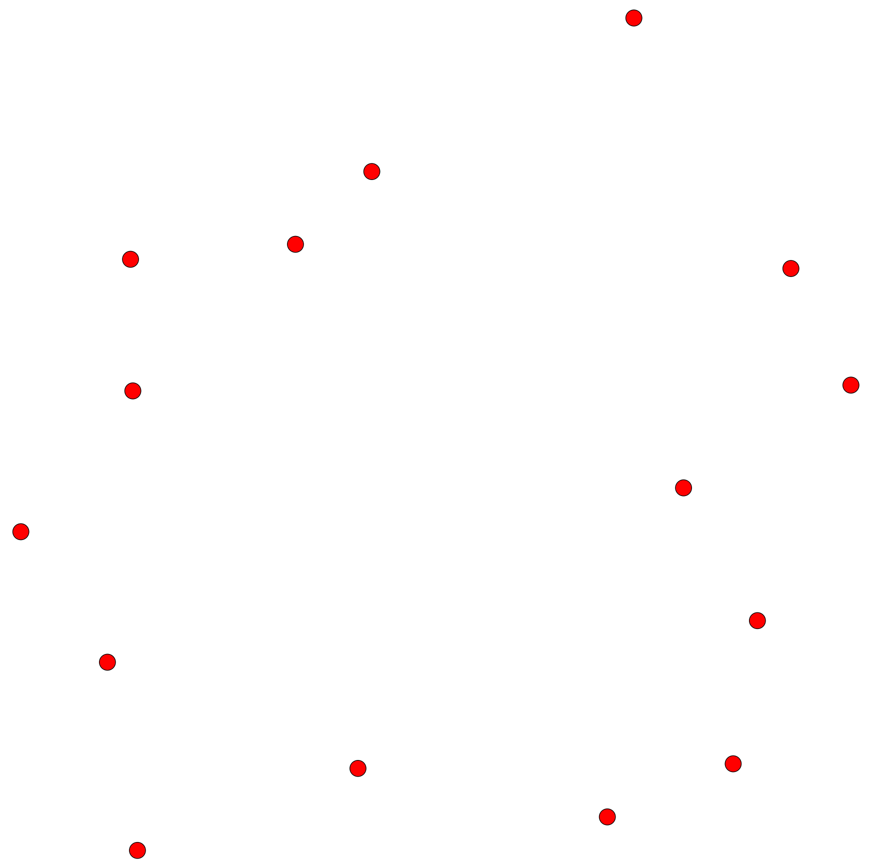
4) C ≔ union of endpoints on the lines



ε-Coreset for 1-Center / Enclosing Balls

Smaller Coreset

5) Return C



ε-Coreset for 1-Center / Enclosing Balls

Smaller Coreset - Proof

C is a coreset for P'

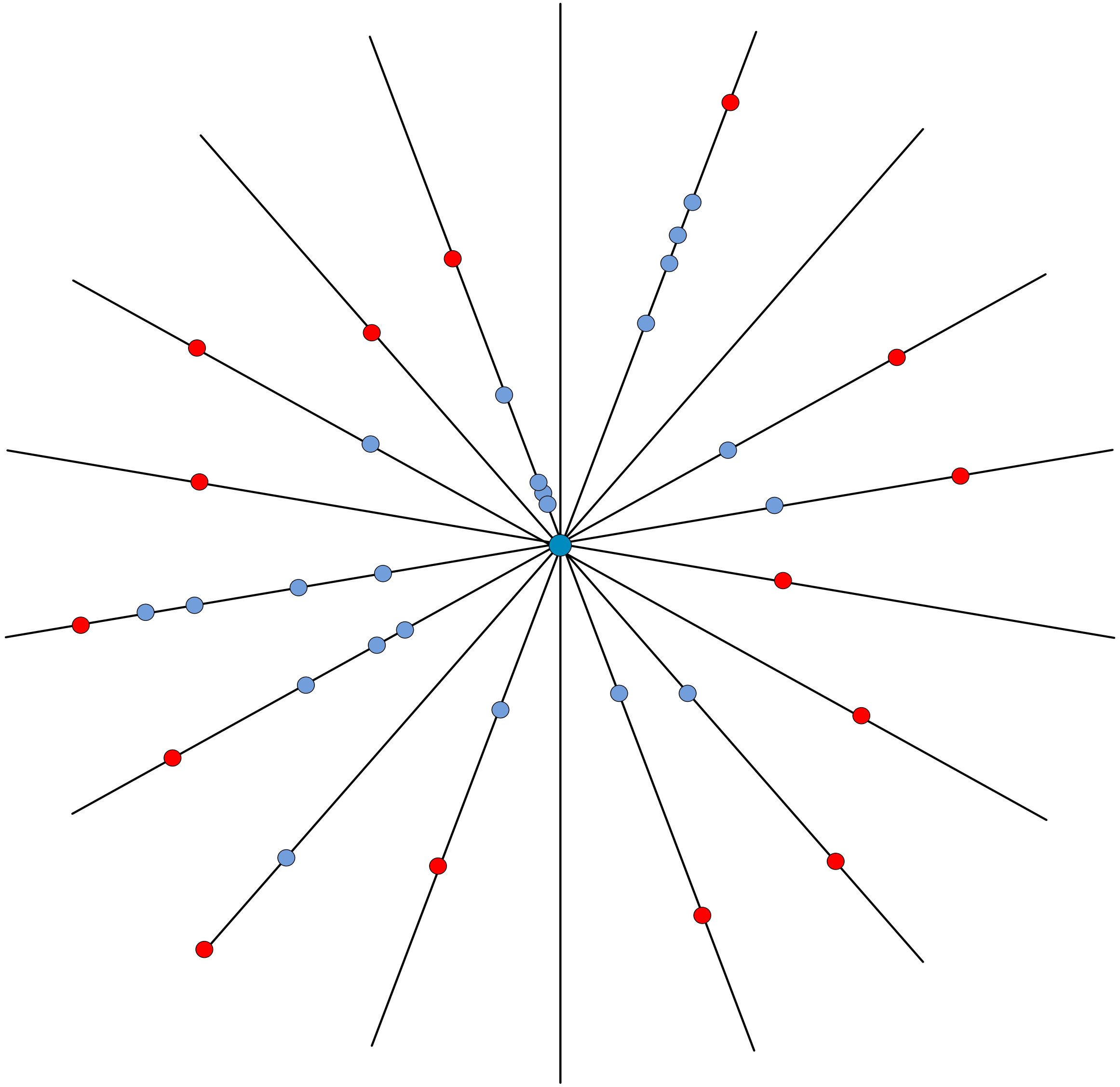
+

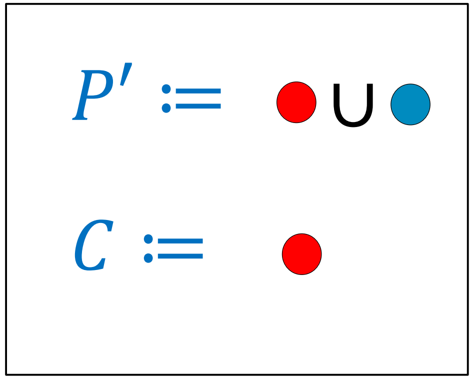
P' is a coreset for P (Large coreset but only few lines)

⬇

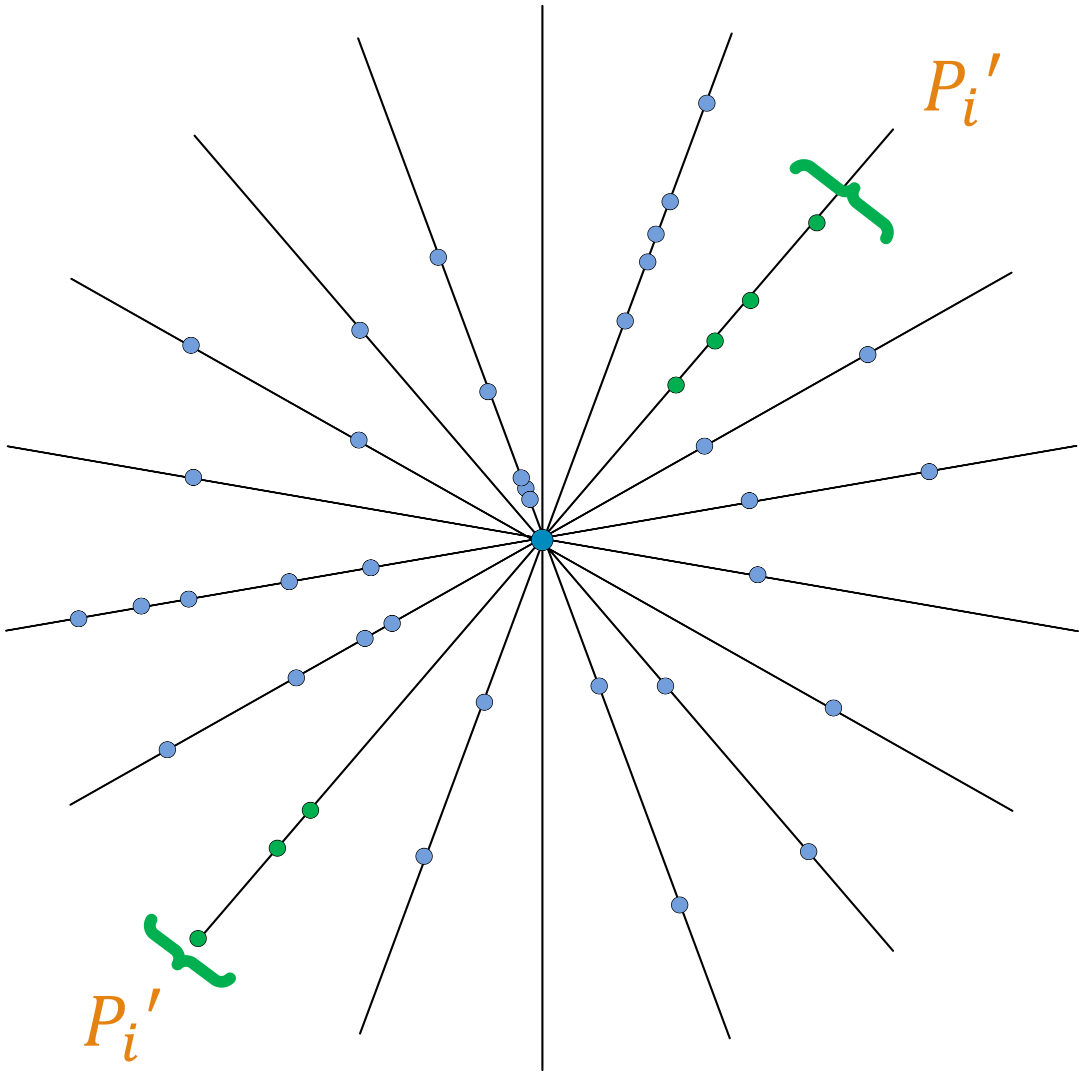
C is a coreset for P (Transitive Property)

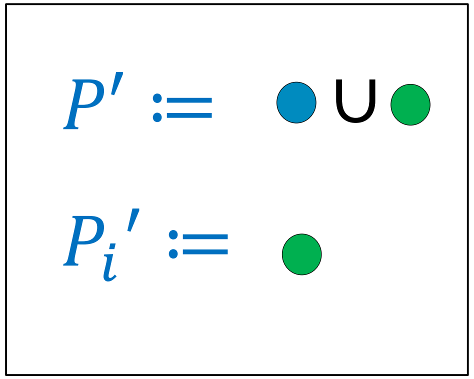
Claim: C is a coreset for P'





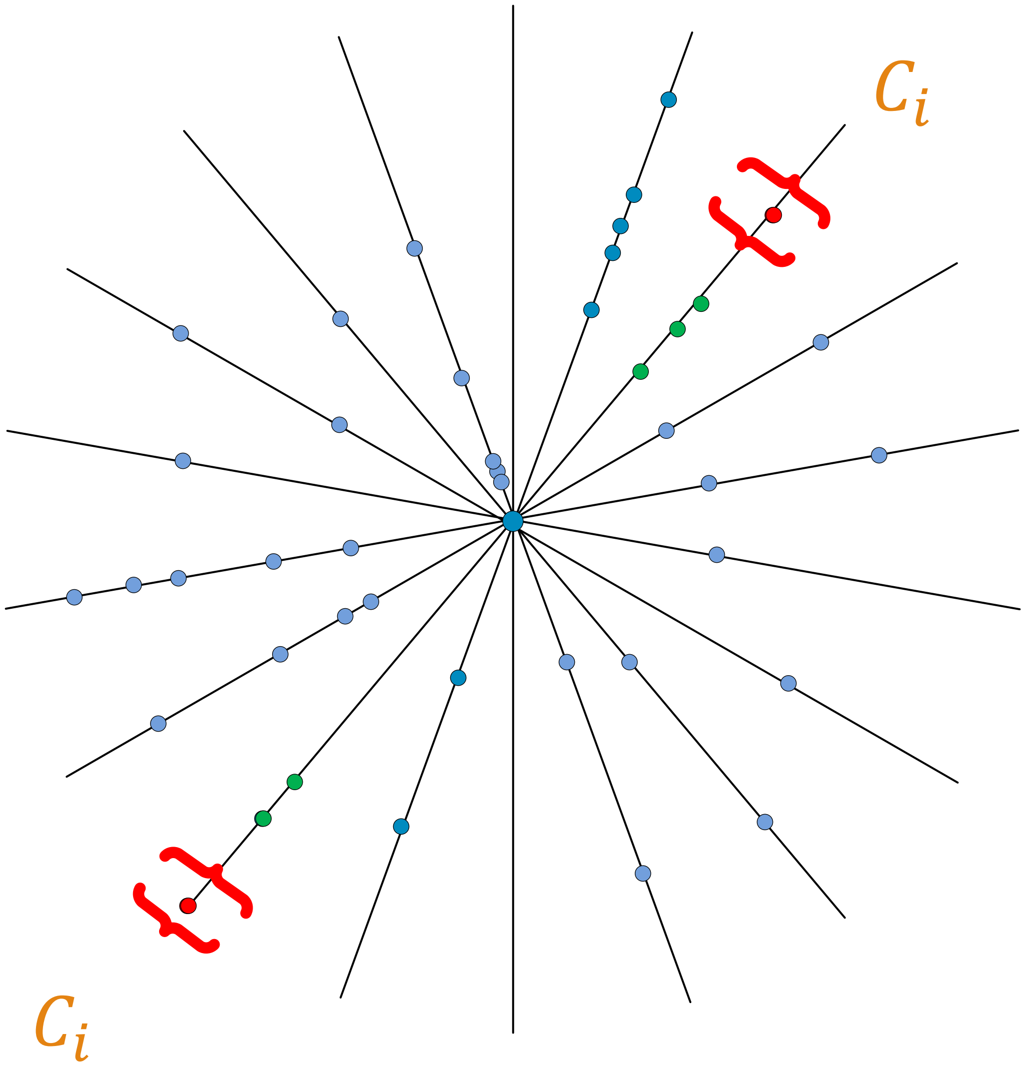
Claim: C is a coreset for P'





$P_i' :=$ intersection of P' with the *i*-th line

Claim: C is a coreset for P'



$P_i' \coloneqq$ (green) ∪ (red)

$C_i \coloneqq$ (red)

$P_i' \coloneqq$ intersection of P' with the *i*-th line

$C_i \coloneqq P' \cap C$

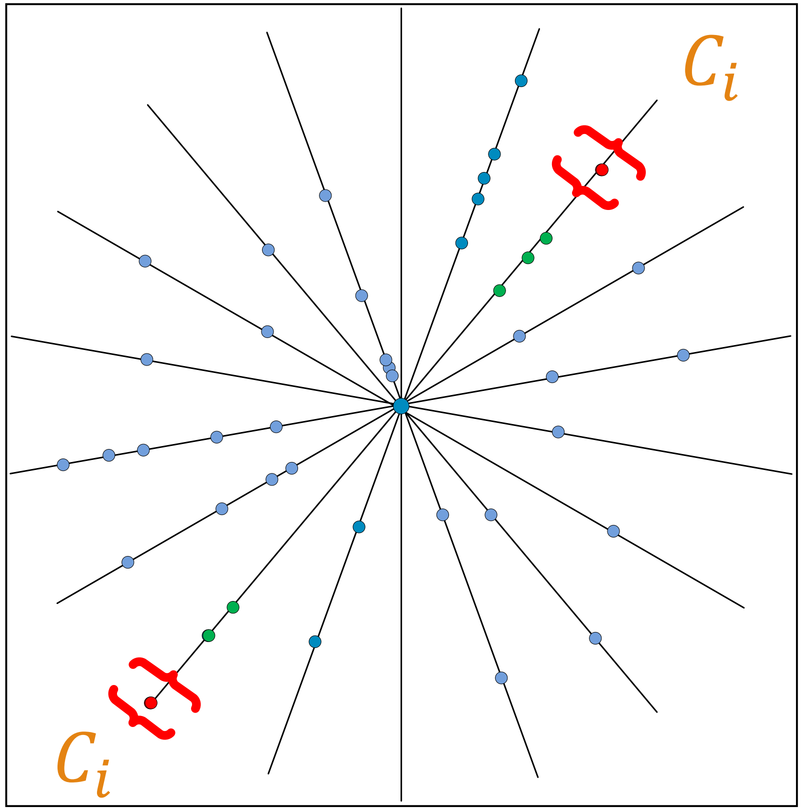
Claim: C is a coreset for P'

$P_i{}' :=$ intersection of P' with the i-th line

$C_i := P' \cap C$

(By proof for d=1)

C_i is a coreset for P_i



Claim: C is a coreset for P'

$P_i{}'$:= intersection of P' with the *i*-th line

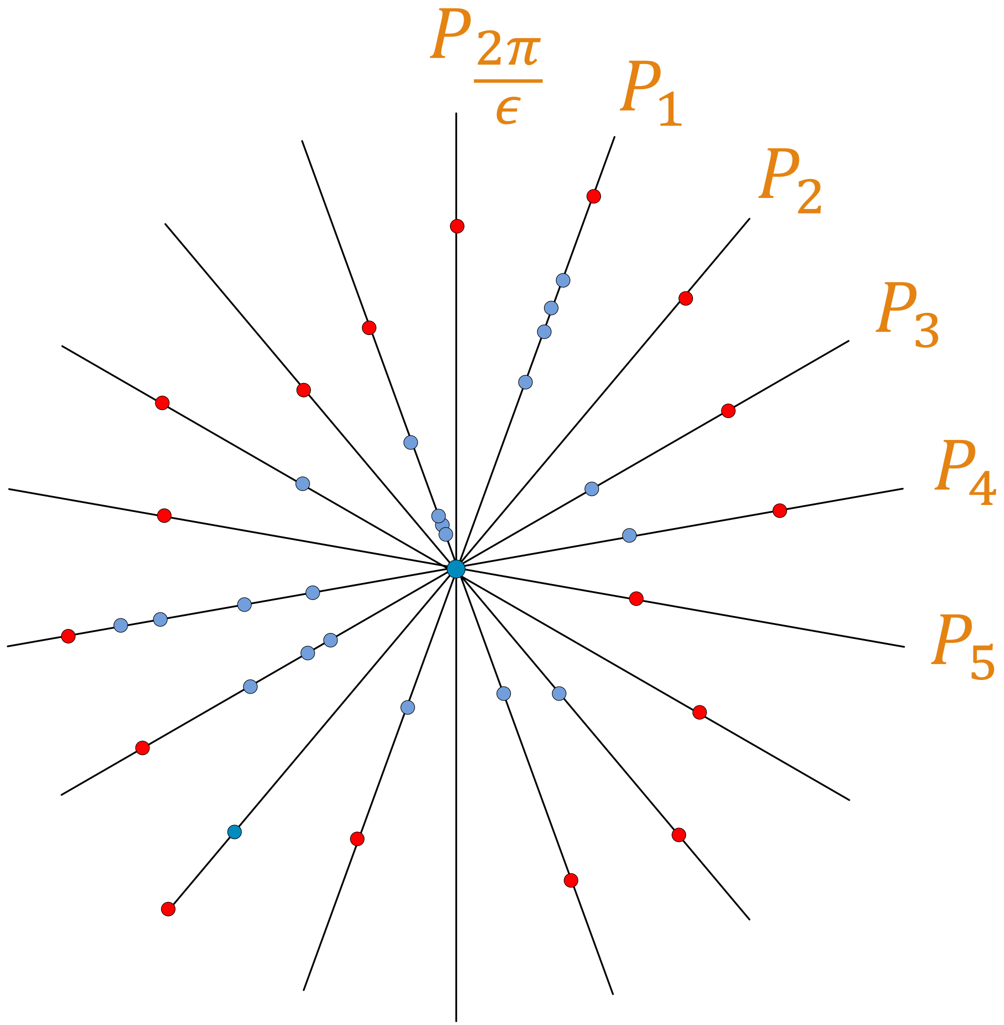
$C_i := P' \cap C$

(By proof for d=1)

C_i is a coreset for P_i

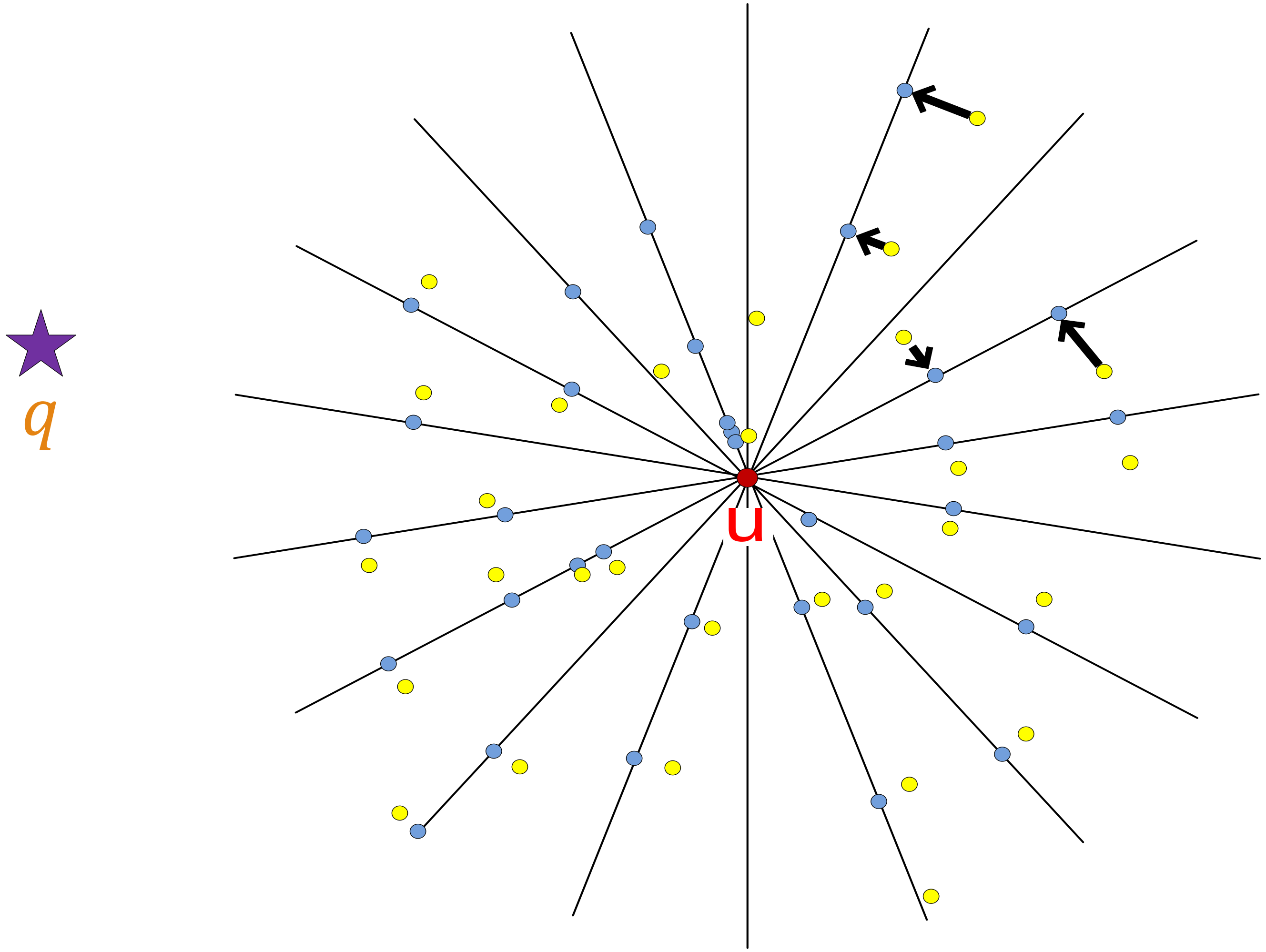
(Union Rule)

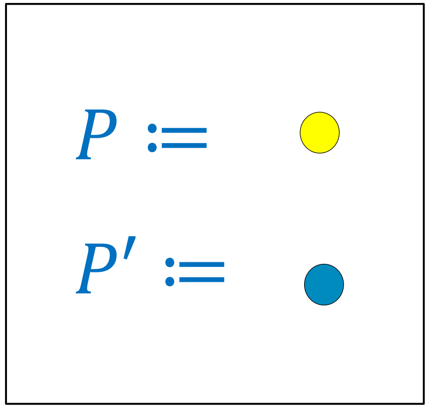
$C := \cup_i C_i$ is a coreset for $P' := \cup_i P_i$



Claim: P' is a coreset for P

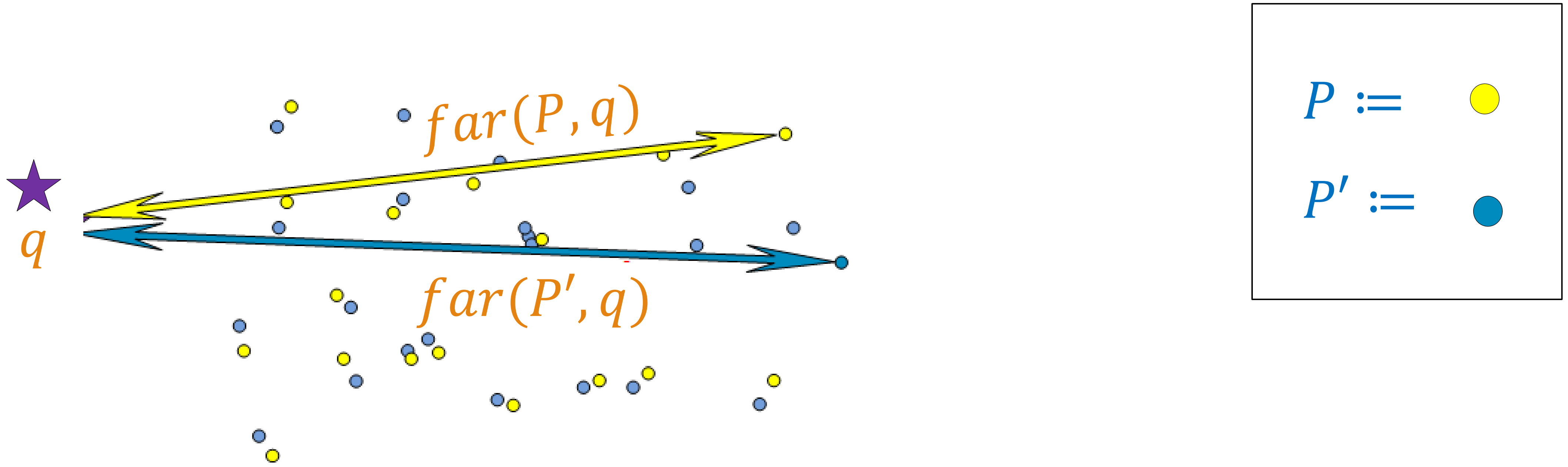
q := an arbitrary query point





Claim: P' is a coreset for P

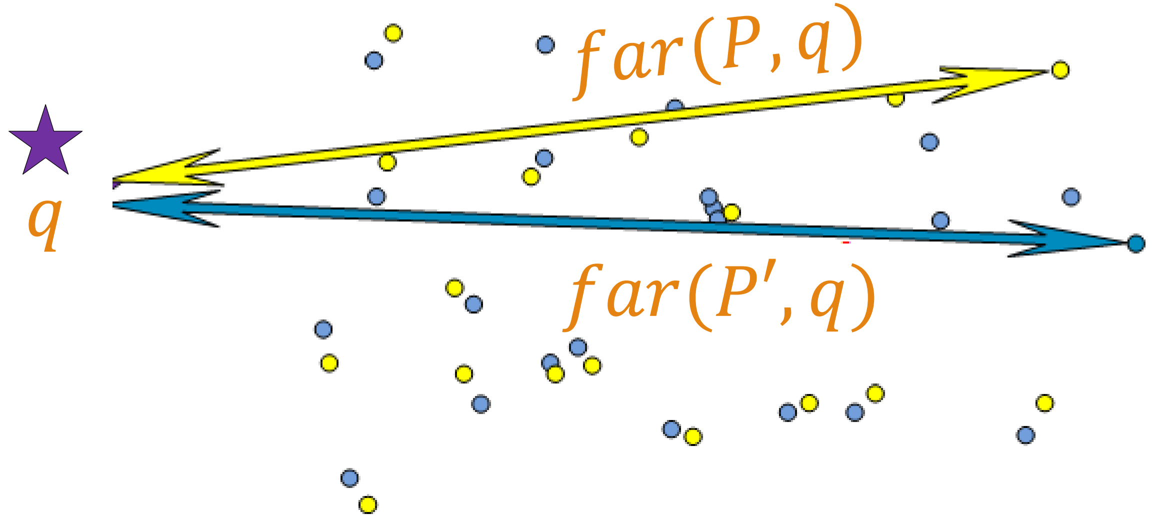
q := an arbitrary query point

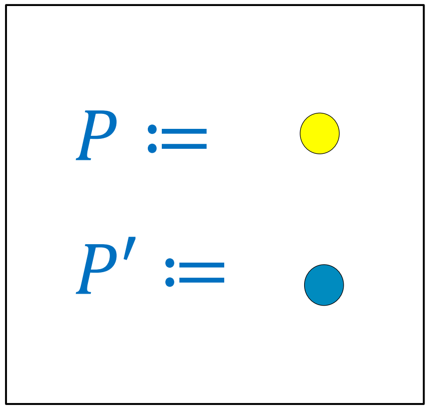


Claim: P' is a coreset for P

q := an arbitrary query point

$C \subseteq P \rightarrow far(C,q) \leq far(P,q)$

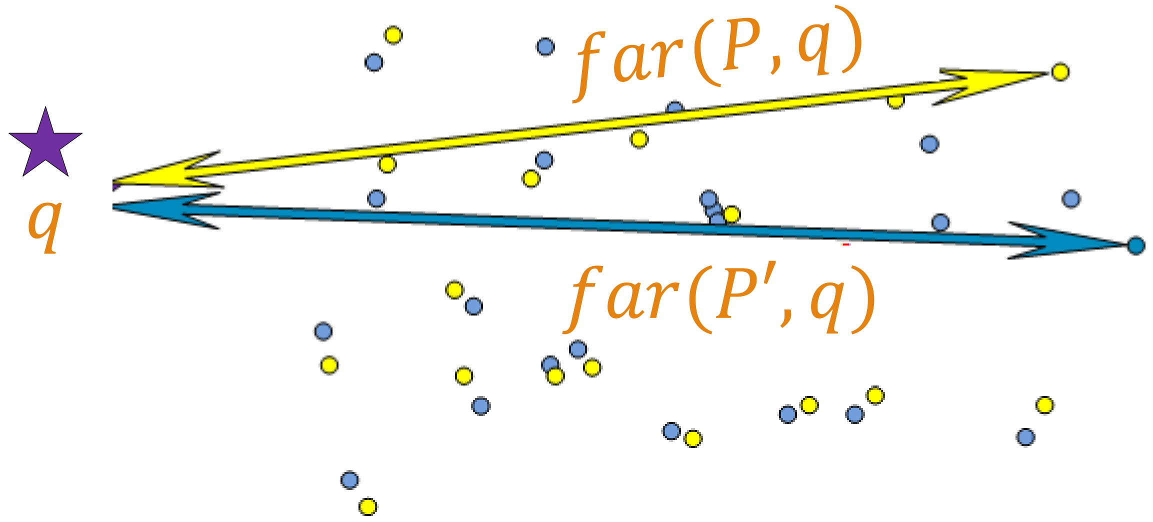


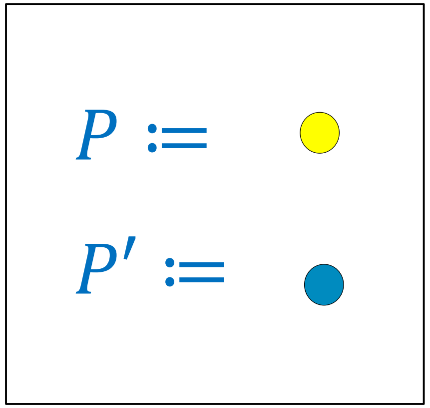


Claim: P' is a coreset for P

Need to prove:

$$far(P,q) - far(P',q) \le \epsilon far(P,q)$$



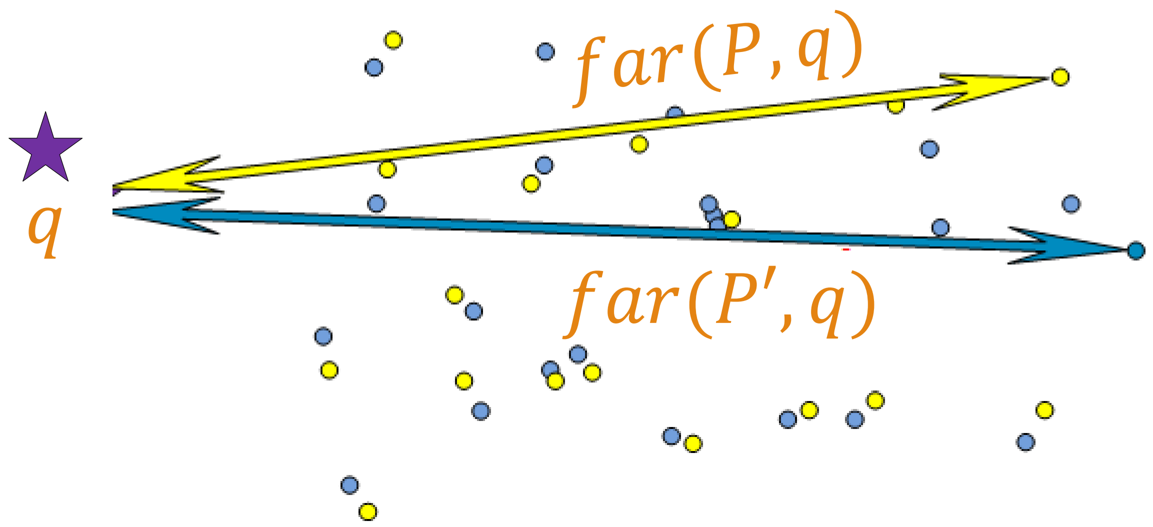


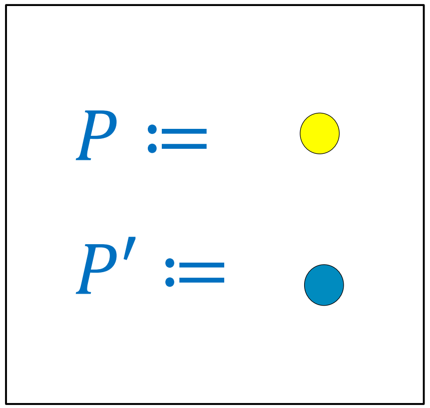
Claim: P' is a coreset for P

Need to prove:

$$far(P,q) - far(P',q) \leq \epsilon far(P,q)$$

$$far(P',q) - far(P,q) \leq \epsilon far(P,q)$$

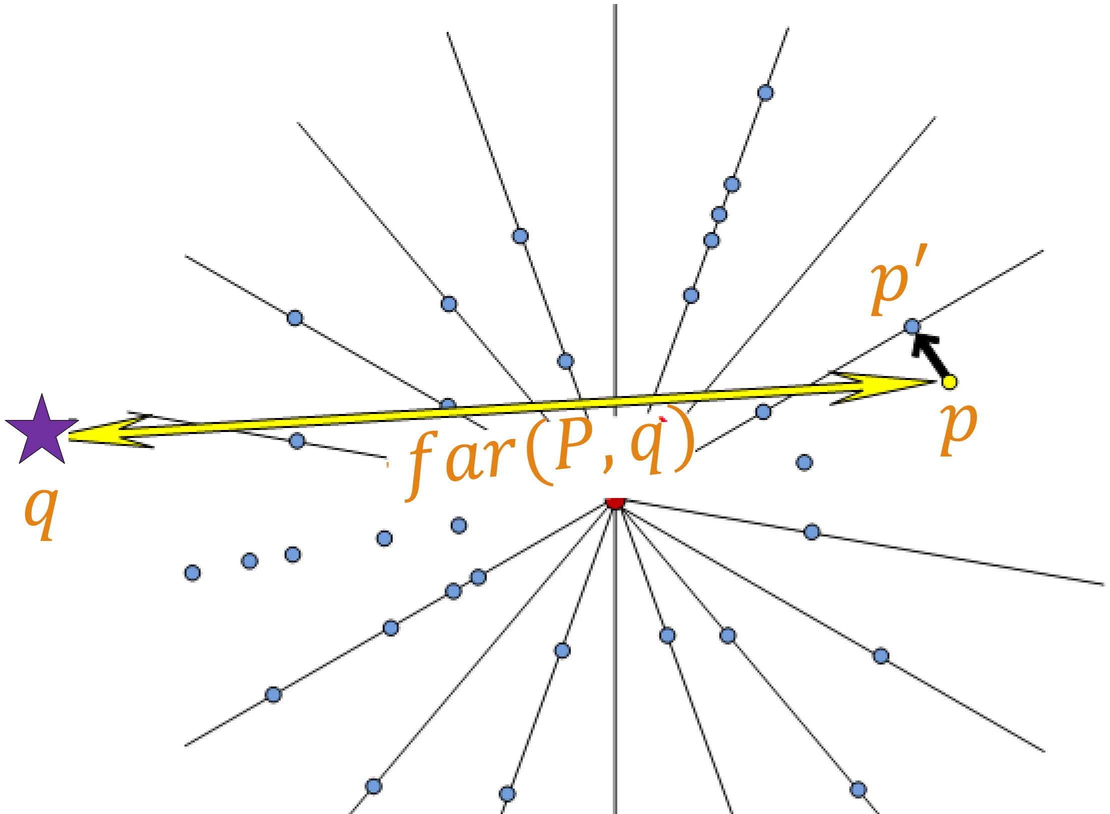




Claim: P' is a coreset for P

Let $far(P, q) = dist(p, q)$

$p' :=$ the projection of p on the "star"

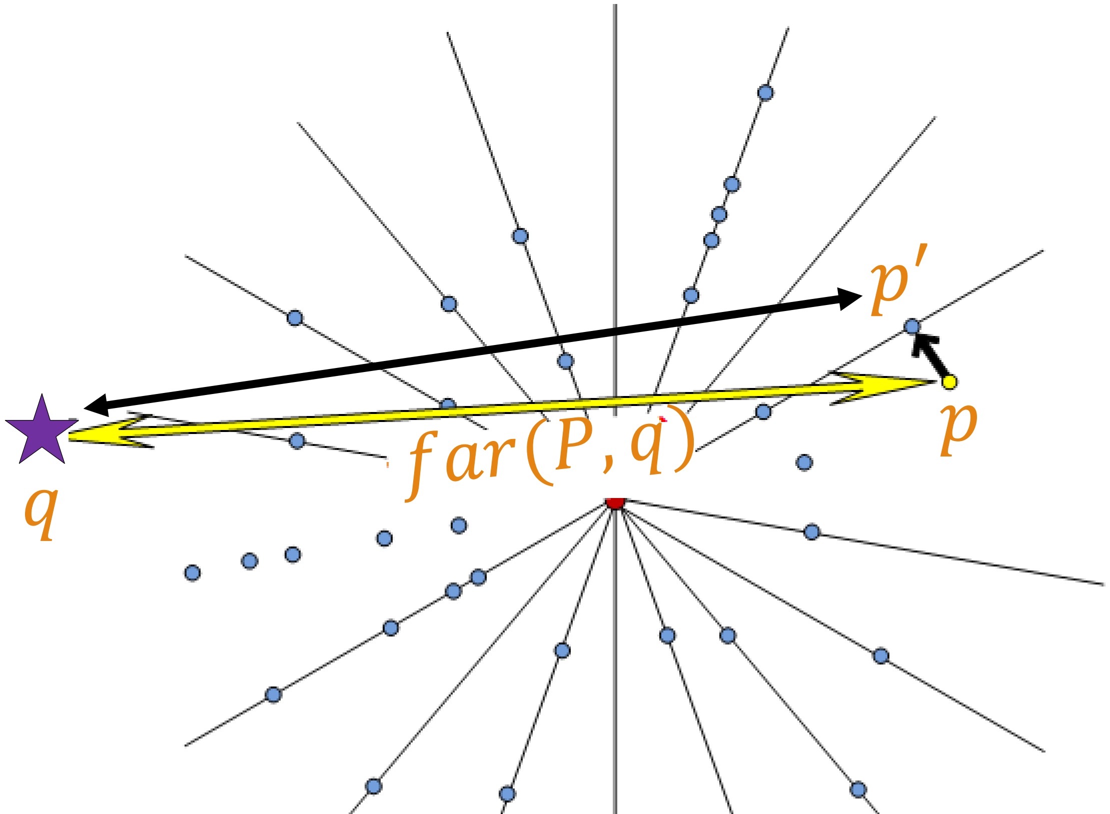


Claim: P' is a coreset for P

Let $far(P,q) = dist(p,q)$

$p' :=$ the projection of p on the "star"

$$far(P,q) - far(P',q)$$
$$\leq far(P,q) - dist(p',q)$$
$$\leq dist(p,p')$$

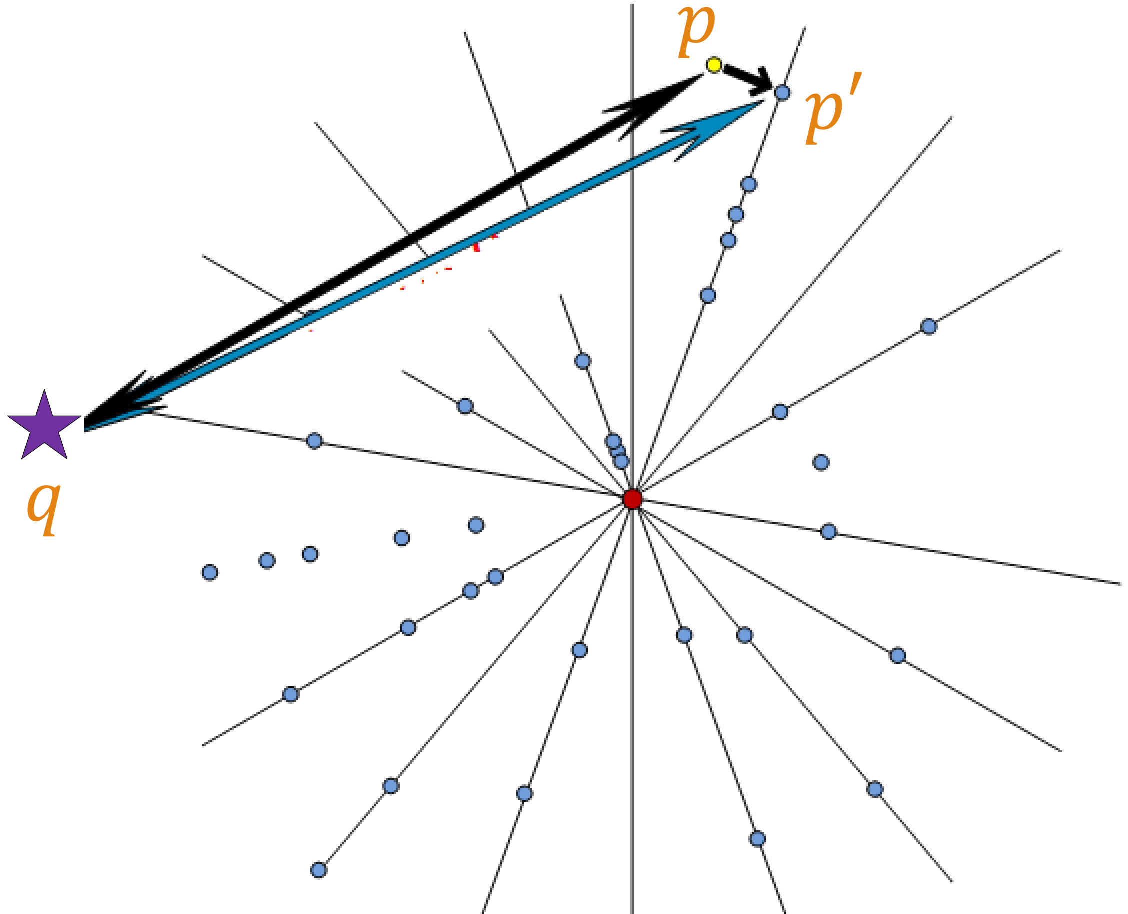


Claim: P' is a coreset for P

Let $far(P', q) = dist(p', q)$

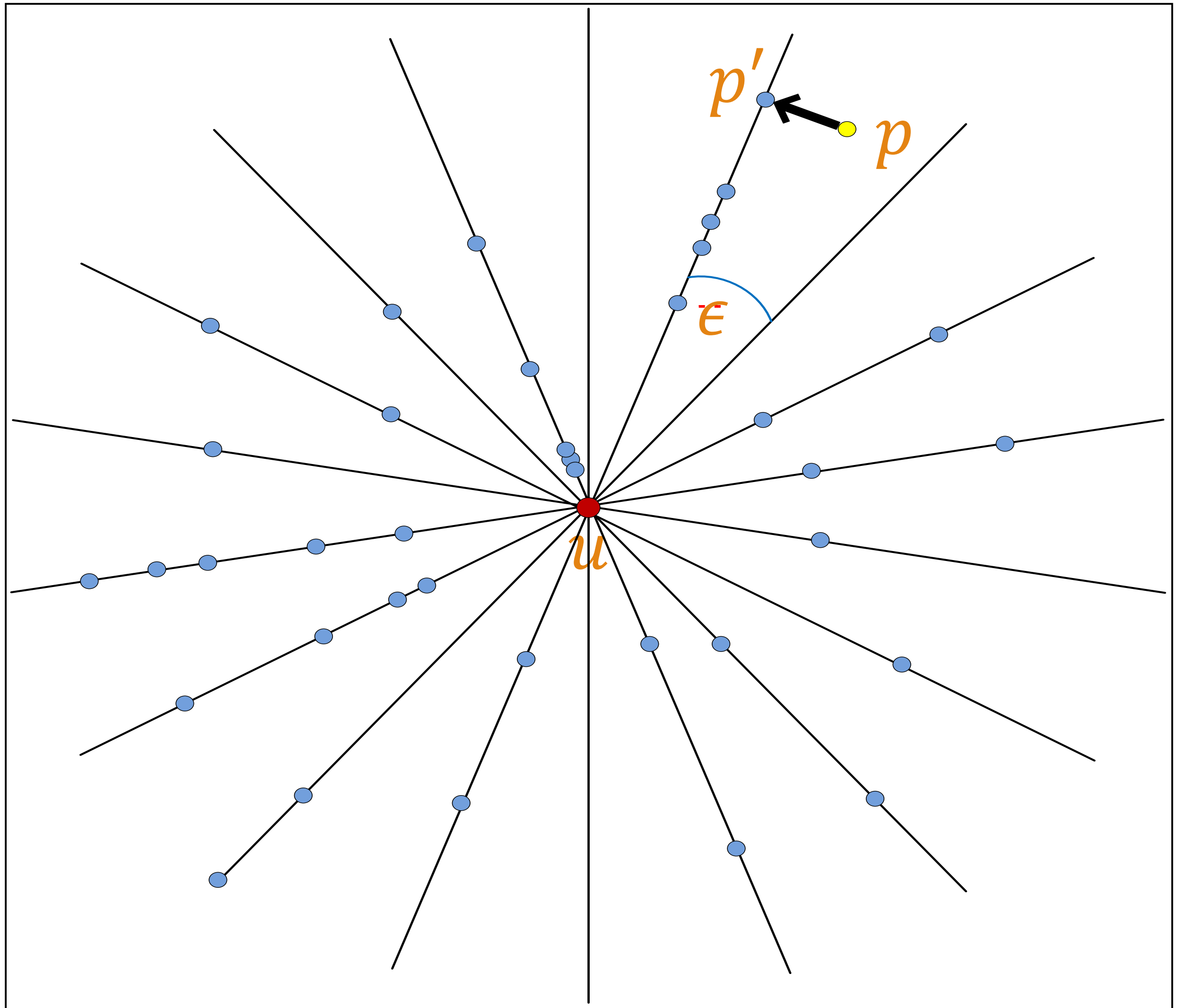
$p :=$ the point whose projection is p'

$$far(P', q) - far(P, q)$$
$$\leq far(P', q) - dist(p, q)$$
$$\leq dist(p, p')$$



Bounding $dist(p, p')$

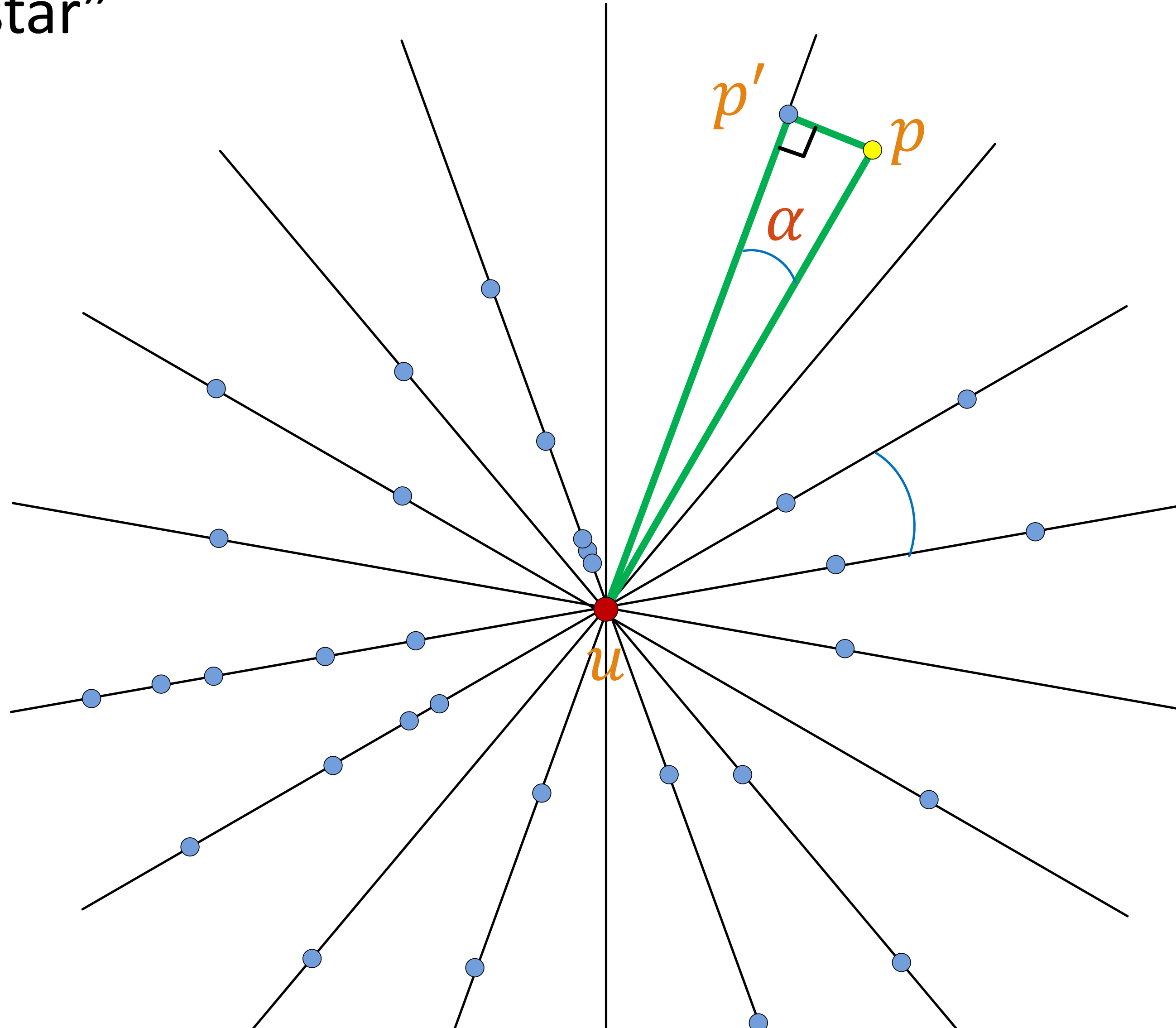
$p' :=$ the projection of p on the "star"



Bounding $dist(p, p')$

$p' :=$ the projection of p on the "star"

$$dist(p, p') = \sin \alpha \cdot dist(u, p)$$
$$\leq O(\epsilon) \cdot dist(u, p)$$

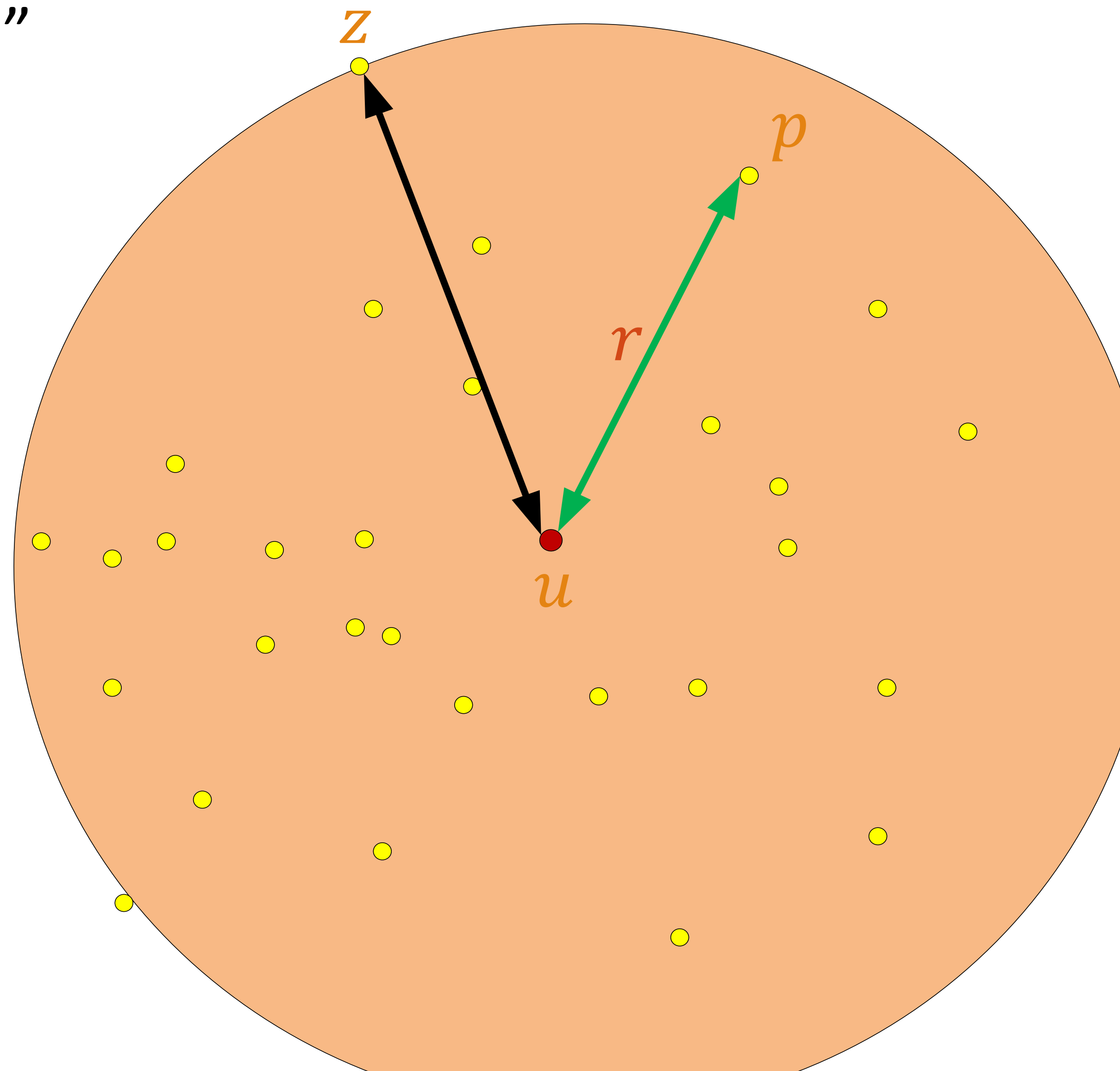


Bounding $dist(p, p')$

p' ≔ the projection of p on the “star”

$$\begin{aligned} dist(p, p') &= \sin\alpha \cdot dist(u, p) \\ &\le O(\epsilon) \cdot dist(u, p) \\ &\le O(\epsilon) \cdot r \end{aligned}$$

$$r \coloneqq \max_{p \in P} dist(u, p)$$



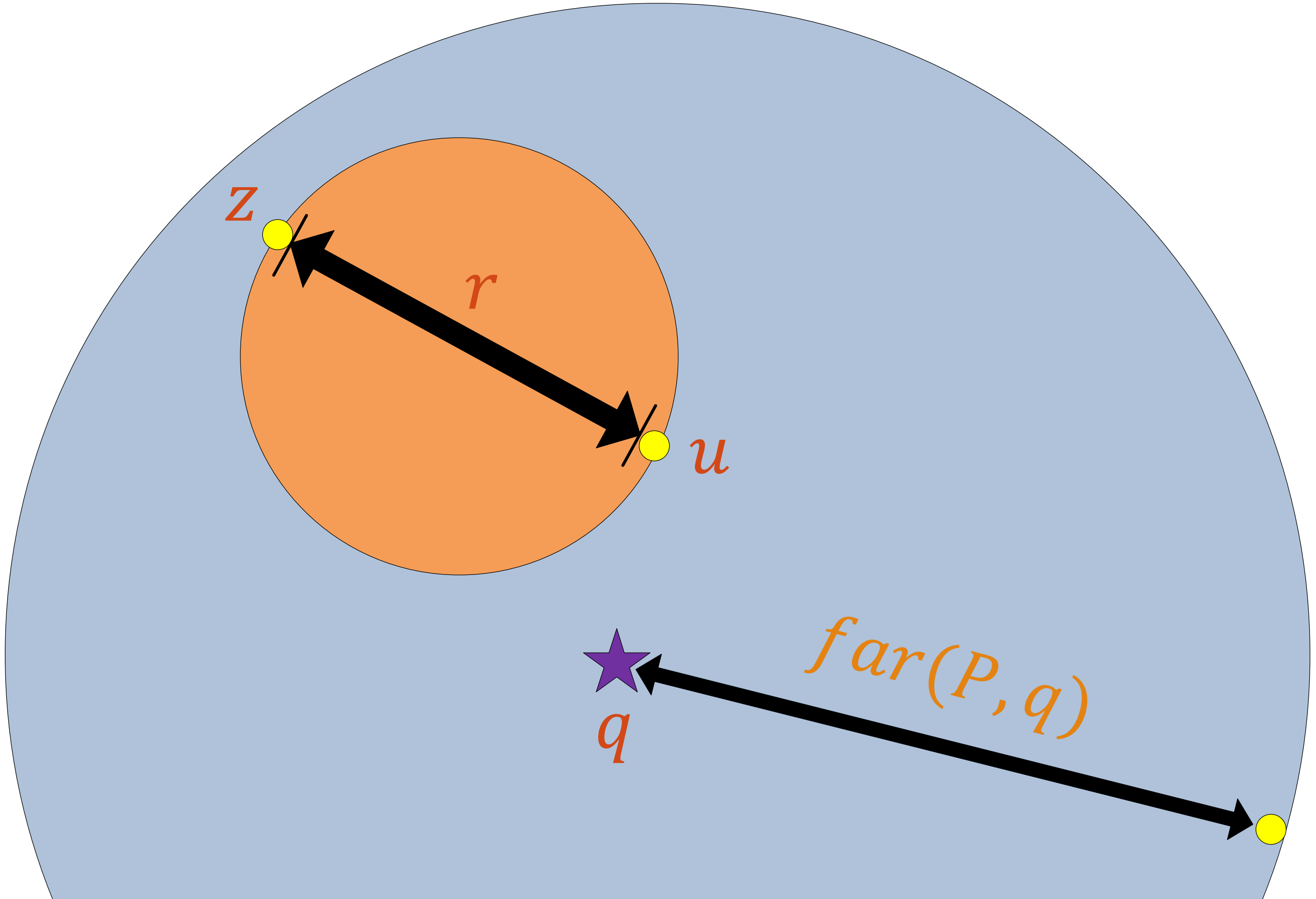
Bounding $dist(p, p')$

Main observation:

Every ball that covers u and z, has a diameter of at least r

$r \leq 2far(P, q)$

$O(\epsilon r) \leq O(\epsilon)far(P, q)$



Bounding $dist(p, p')$

$p' :=$ the projection of p on the "star"

$$\begin{aligned} dist(p, p') &= \sin\alpha \cdot dist(u, p) \\ &\le O(\epsilon) \cdot dist(u, p) \\ &\le O(\epsilon) \cdot r \\ &\le O(\epsilon) \cdot far(P, q) \end{aligned}$$

$$r := \max_{p \in P} dist(u, p)$$