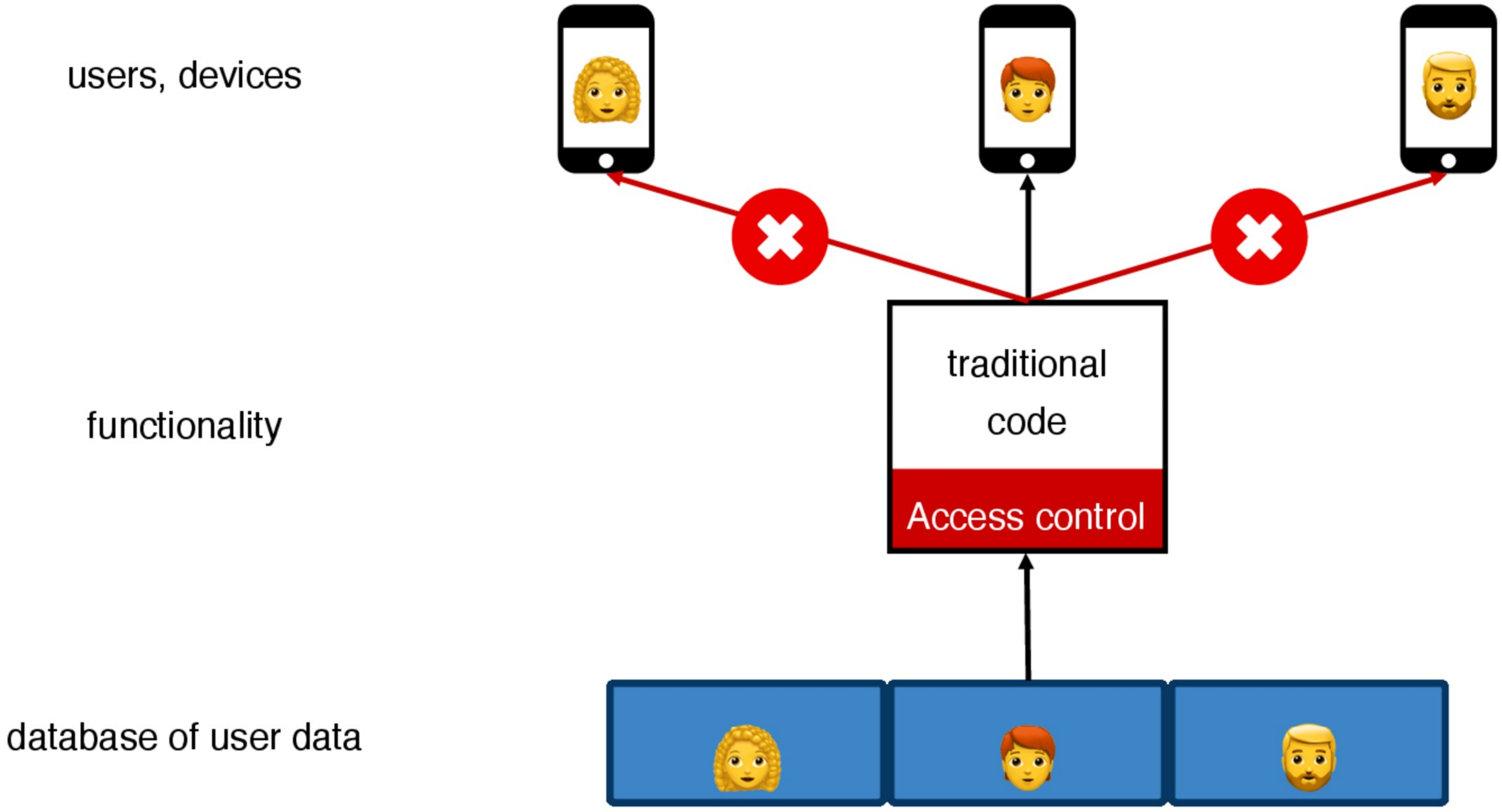
Privacy Budget Scheduling

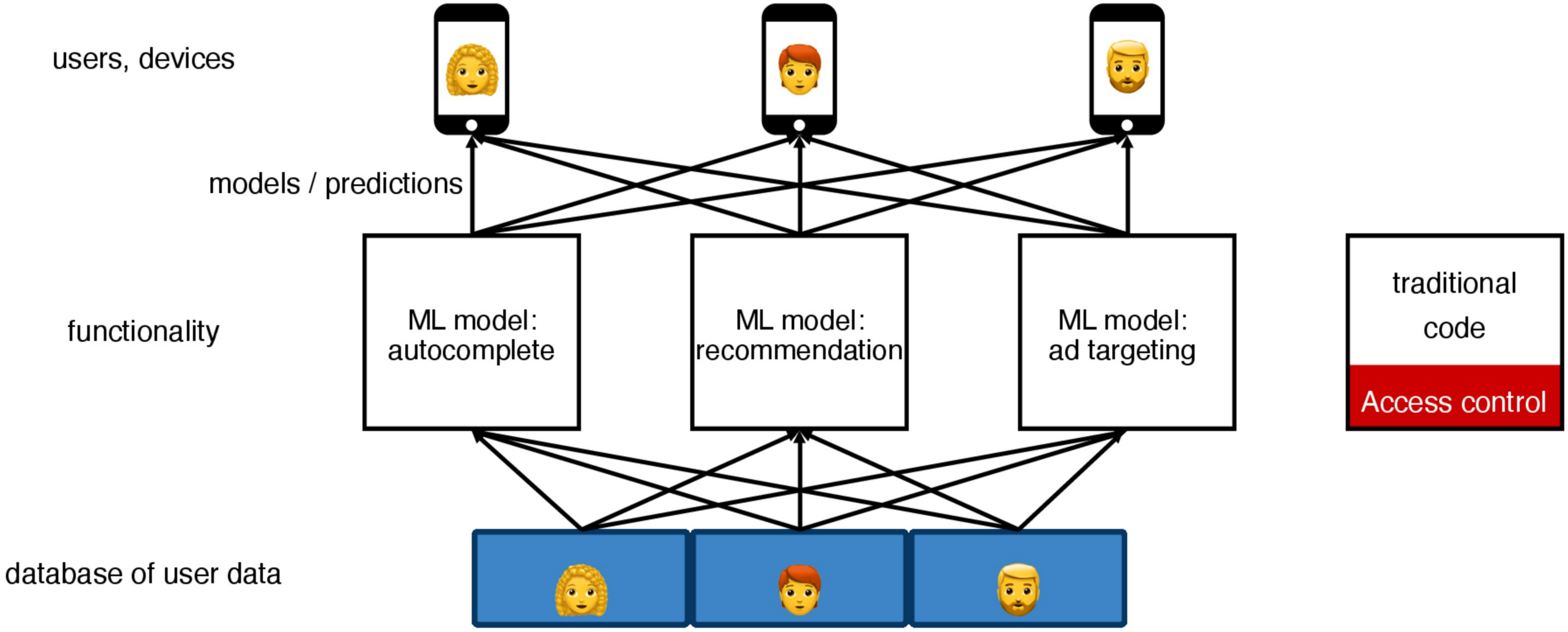
Tao Luo*, Mingen Pan*, **Pierre Tholoniat***
Asaf Cidon*, Roxana Geambasu*, Mathias Lécuyer†

**Columbia University* †*Microsoft Research*

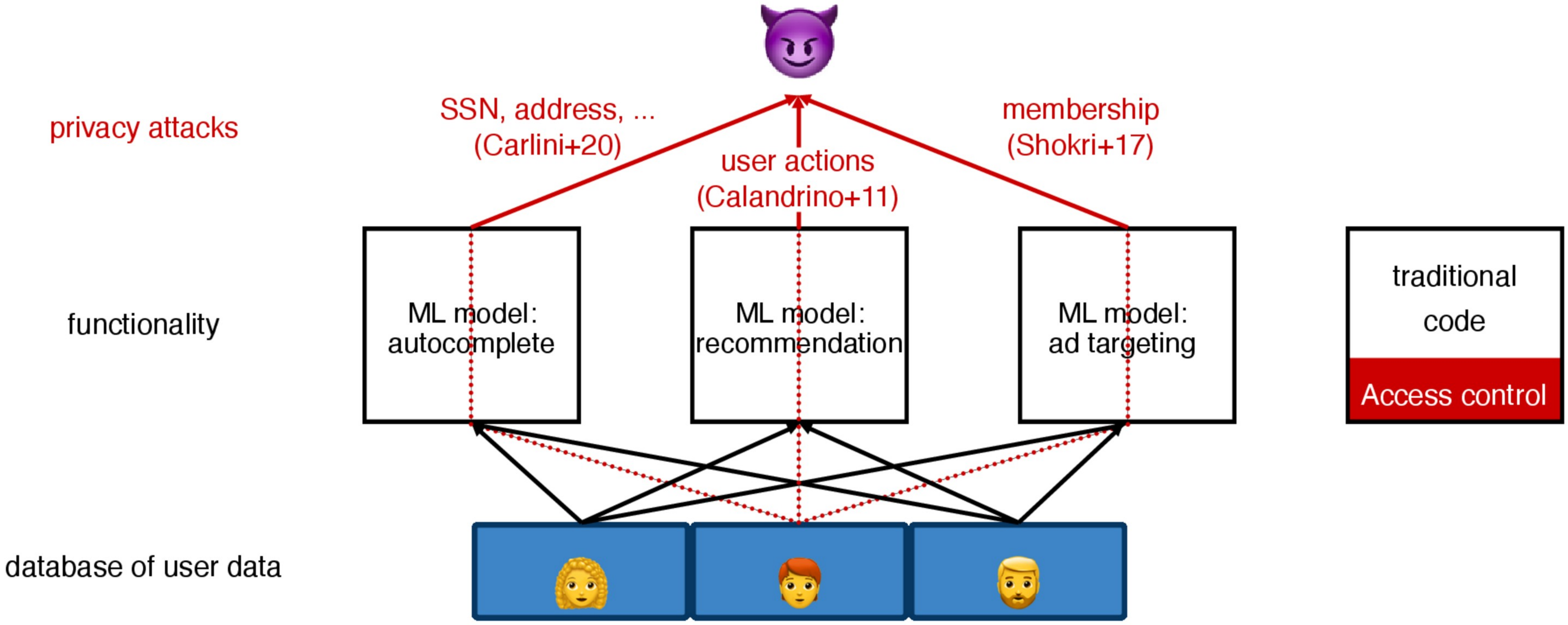
Example: Messaging App



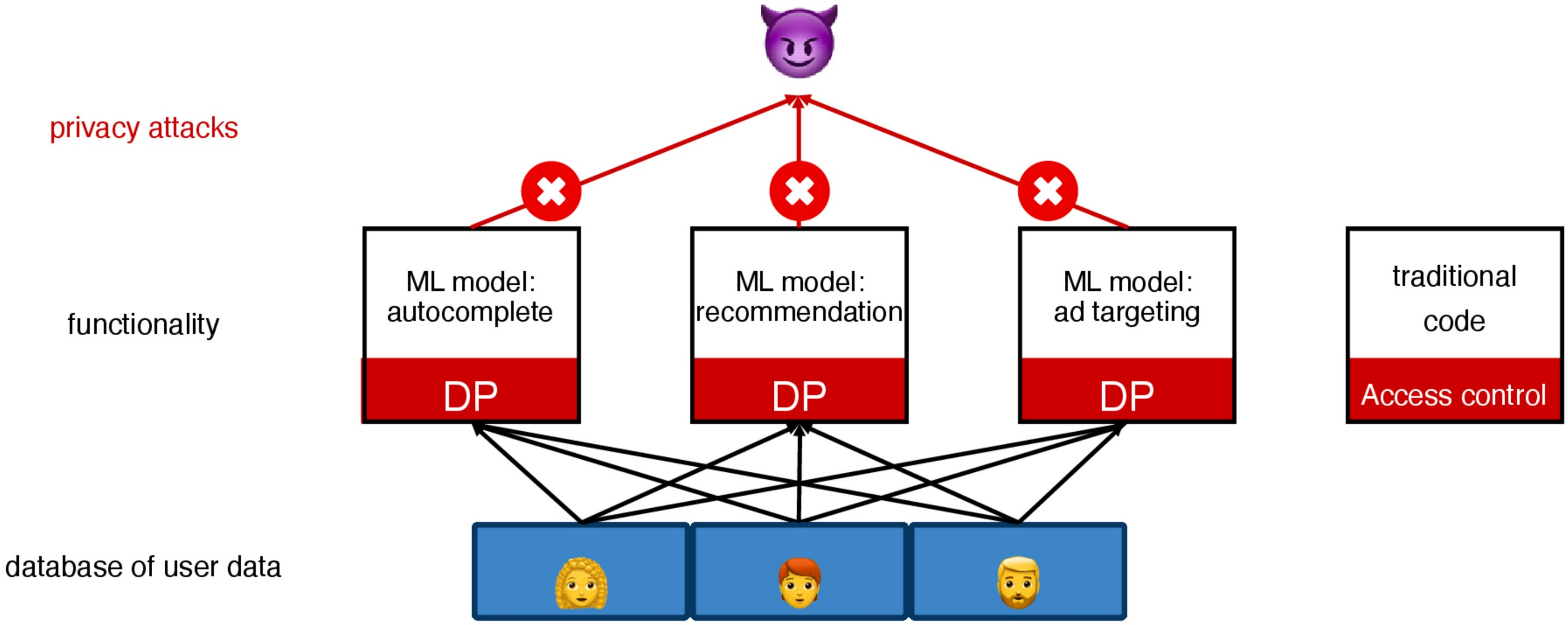
Example: Messaging App



What Can Leak?

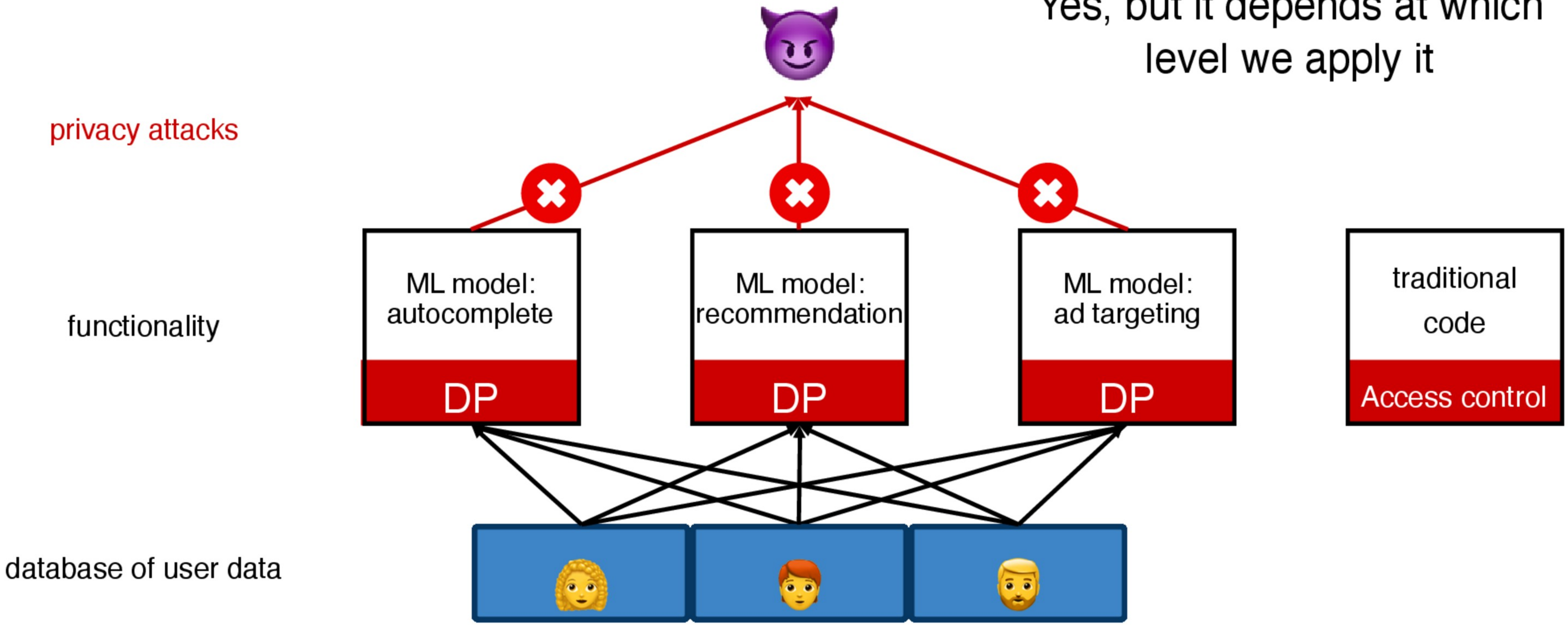


Is Differential Privacy (DP) the Solution?



Is Differential Privacy (DP) the Solution?

Yes, but it depends at which level we apply it



DP at Individual Model Level

- Privacy attacks find data points that make a given observed model more likely

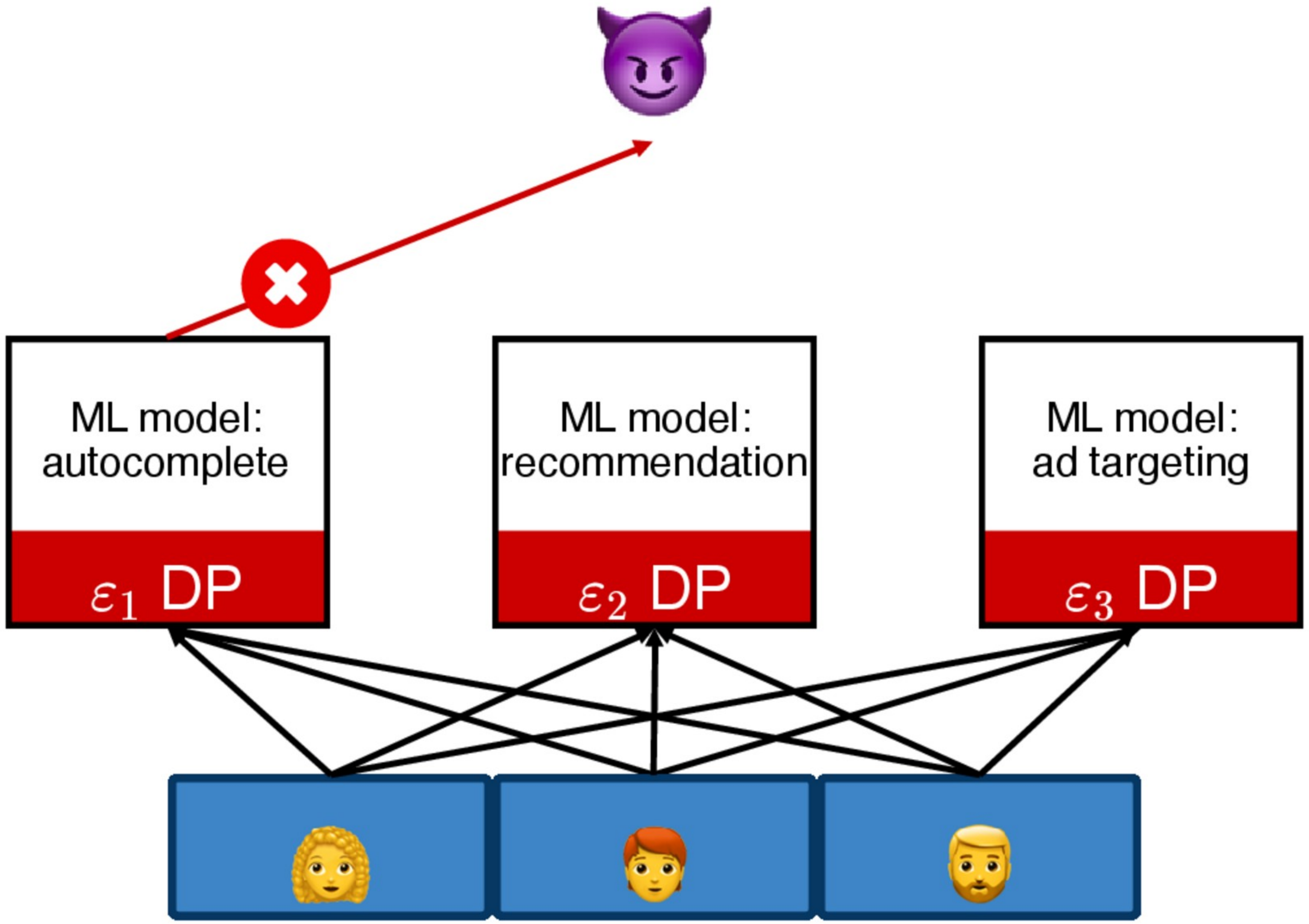
- DP randomizes the training procedure of a model (e.g., SGD) to guarantee that no data point **drastically increases** the likelihood of the outputted model.

- The increase in likelihood of the outputted model is controlled by the **privacy loss** $\varepsilon > 0$

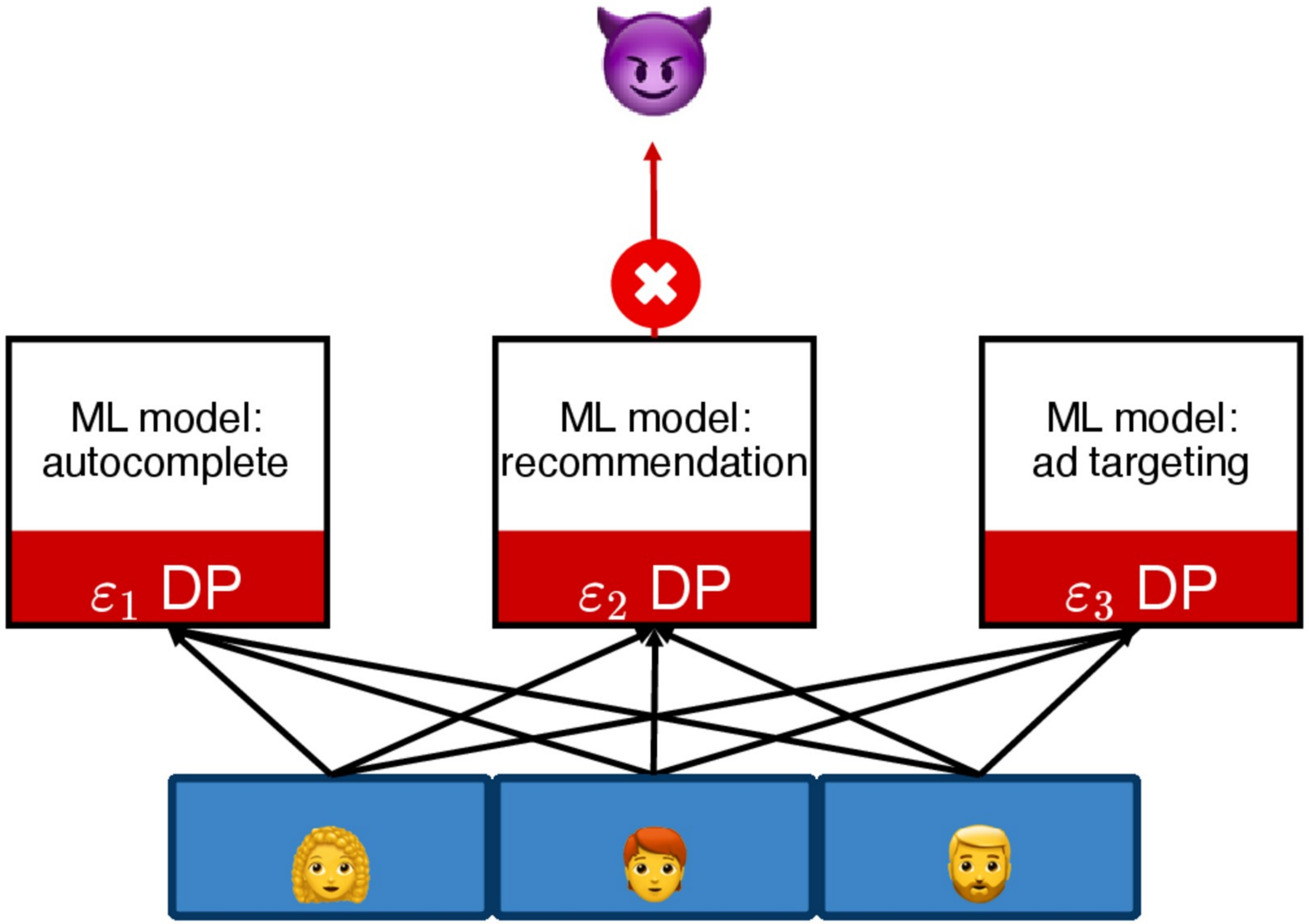
Definition. A randomized procedure $f : D \to O$ over databases is (ε, δ)-**differentially private** if for all databases $d_1, d_2 \in D$ that differ in one data point, and for all output sets $S \subseteq O$:

$$\Pr[f(d_1) \in S] \leq e^{\varepsilon} \Pr[f(d_2) \in S] + \delta$$

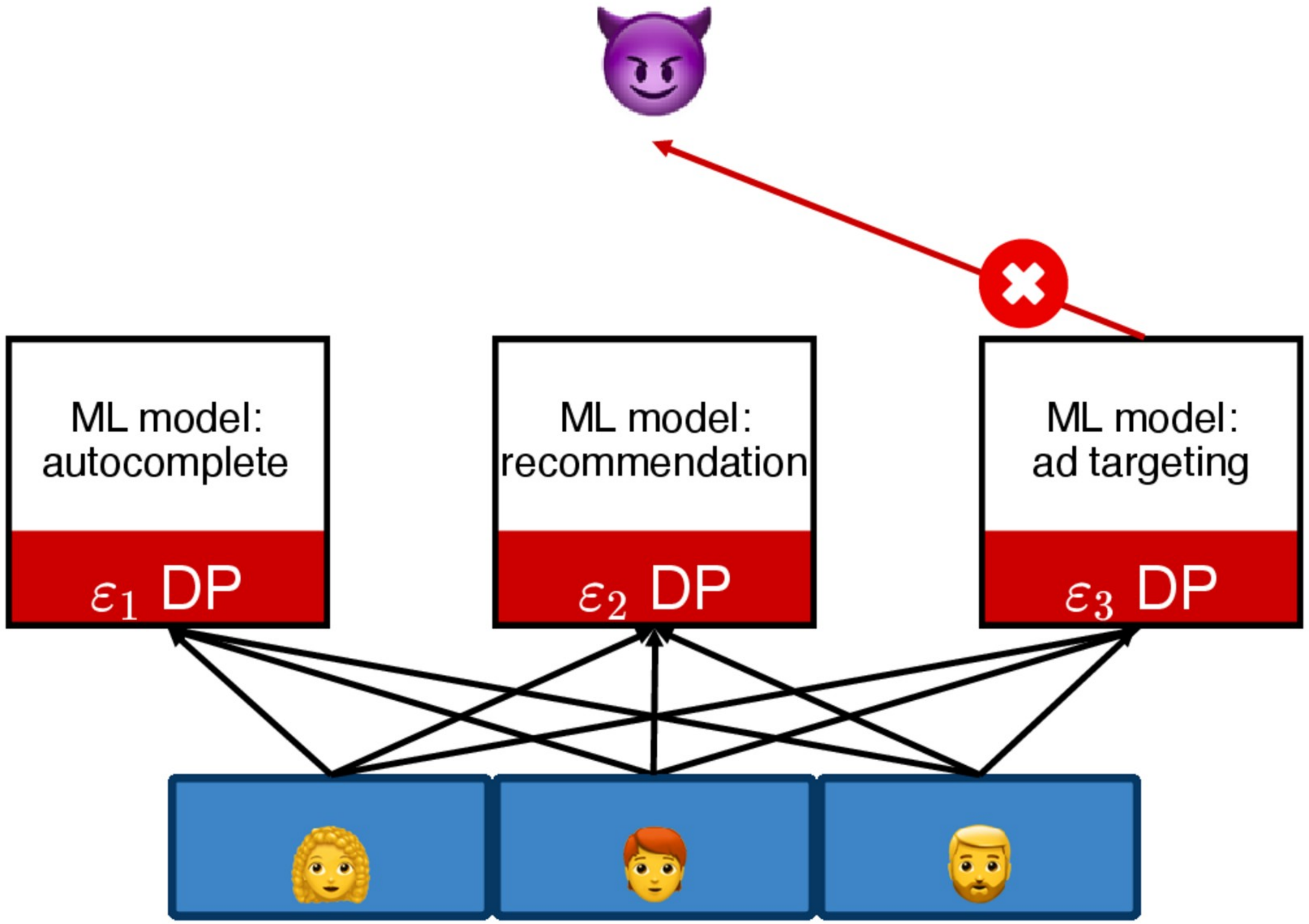
Problem: Privacy Loss Accumulates



Problem: Privacy Loss Accumulates



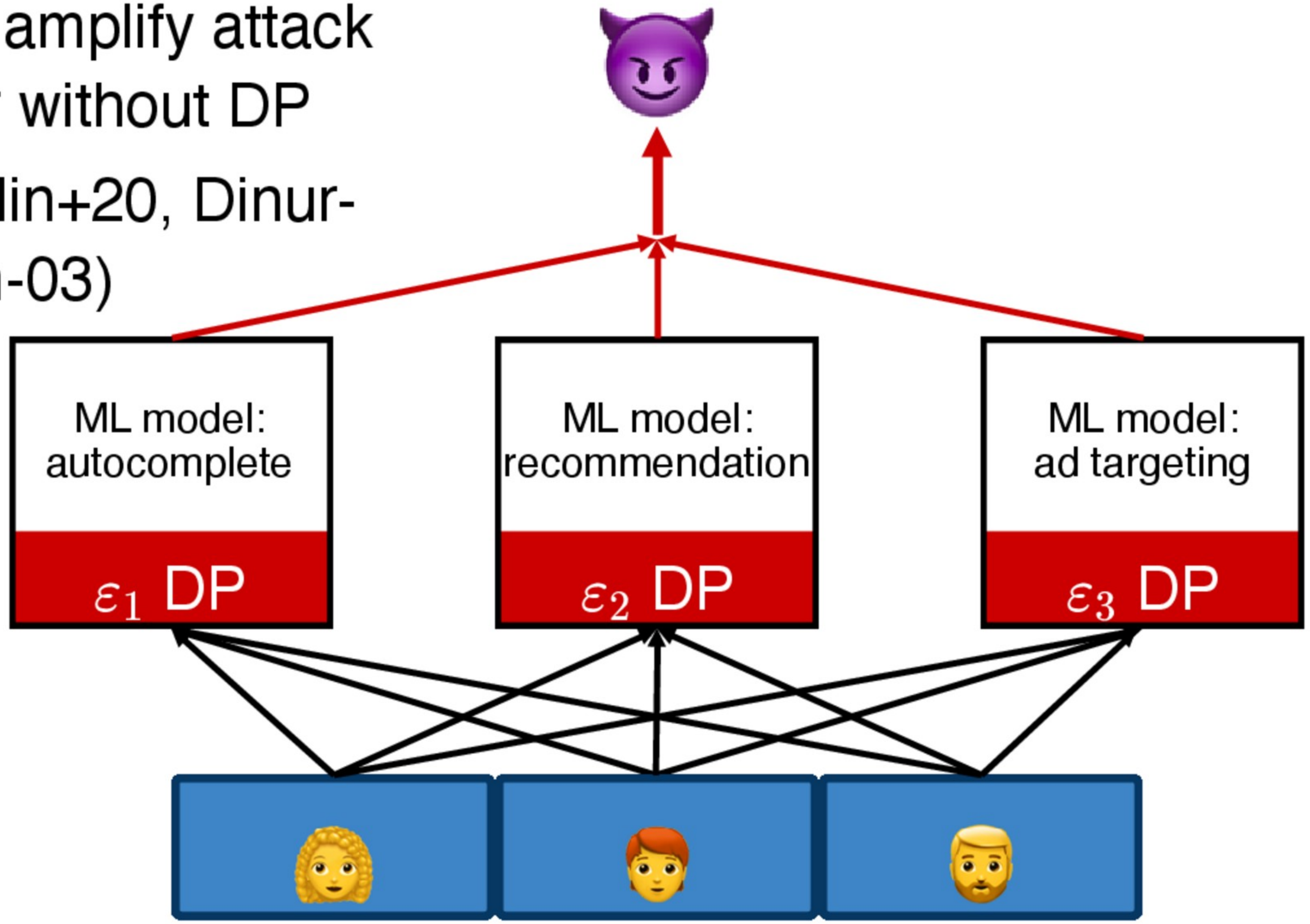
Problem: Privacy Loss Accumulates



Problem: Privacy Loss Accumulates

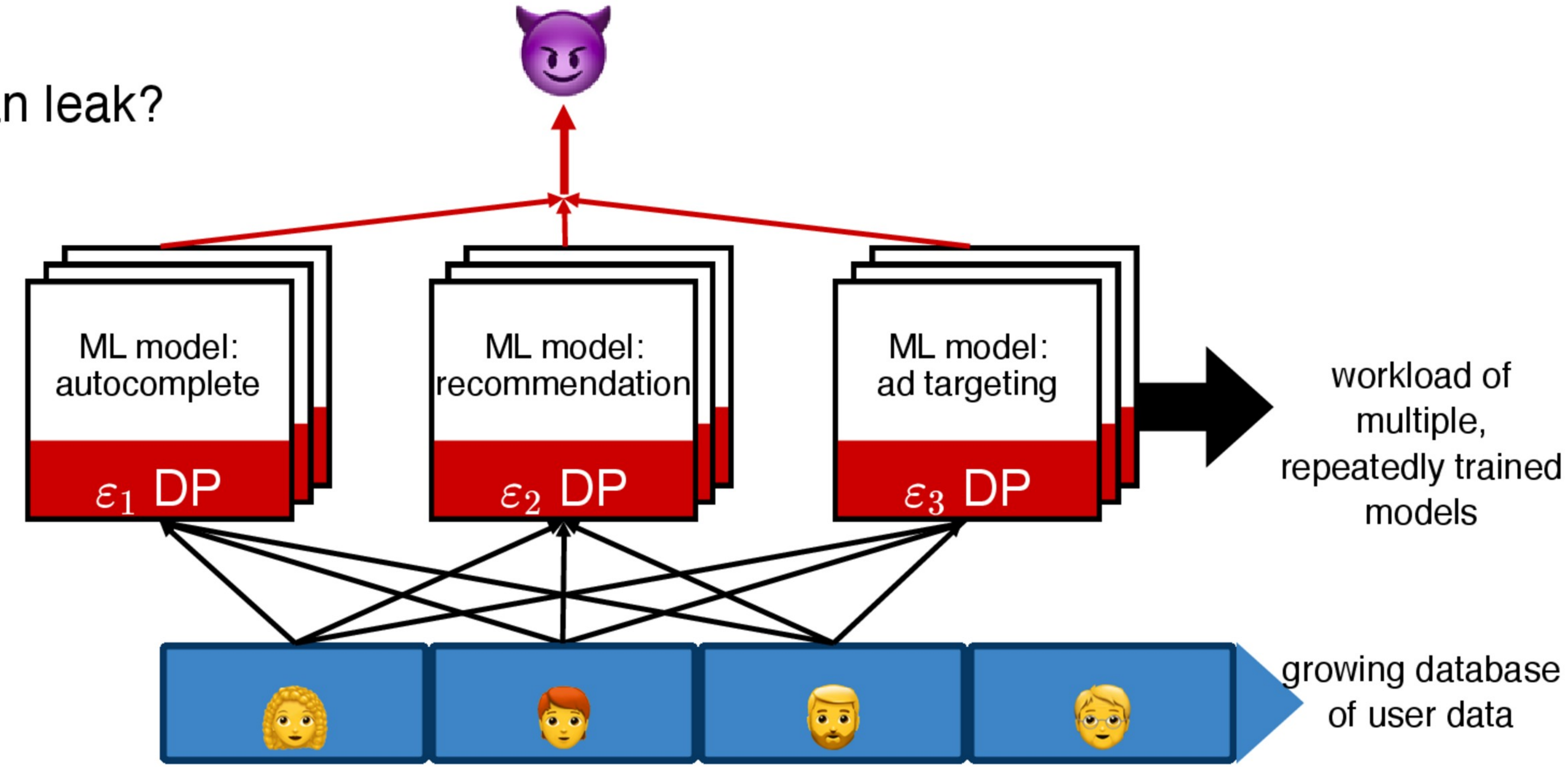
Multiple models amplify attack power, with or without DP

(Zanella-Béguelin+20, Dinur-Nissim-03)



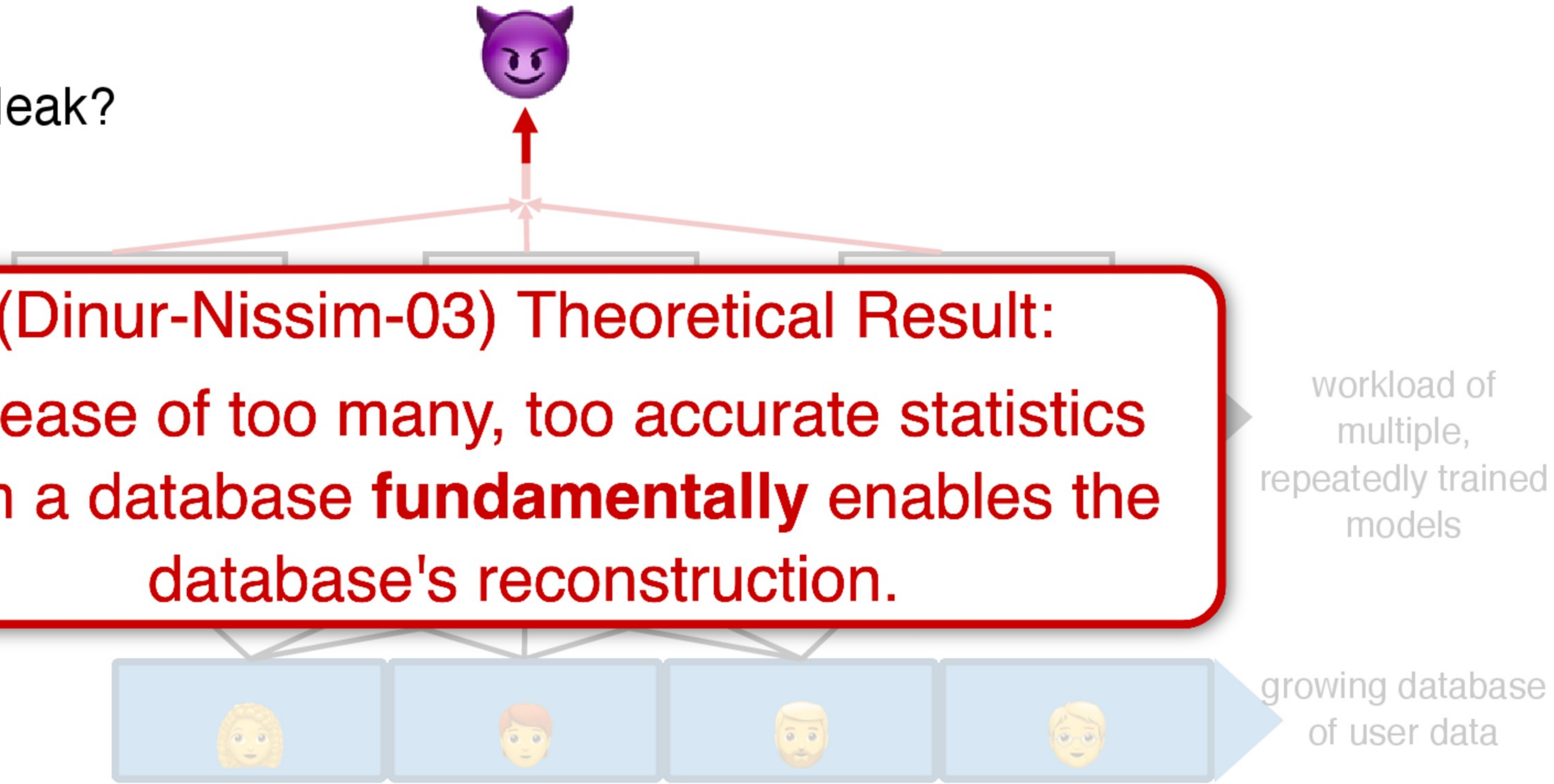
Problem: Privacy Loss Accumulates

What can leak?



Problem: Privacy Loss Accumulates

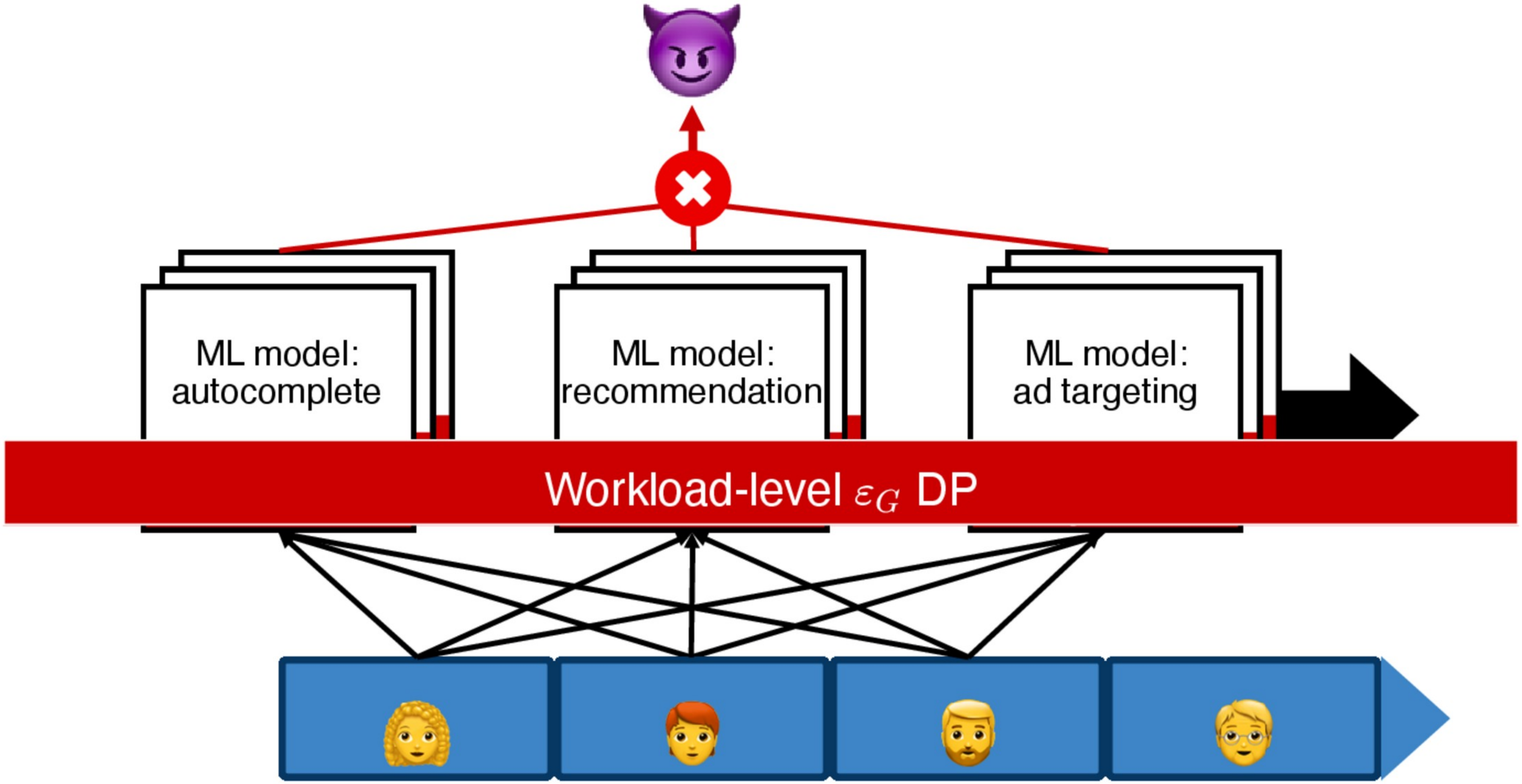
What can leak?



(Dinur-Nissim-03) Theoretical Result:

Release of too many, too accurate statistics from a database **fundamentally** enables the database's reconstruction.

Solution: DP at Workload Level



Our Vision:
Privacy as a Compute Resource

- DP composes, so ML training tasks consume a **global privacy budget** ε_G :

$$\sum_{task\ i} \varepsilon_i \leq \varepsilon_G.$$

- Privacy should be a **compute resource**, alongside CPU, GPU, RAM

- We must **schedule privacy efficiently and fairly**:
  - Can we use existing schedulers? Which ones?
  - Which fairness/efficiency properties?

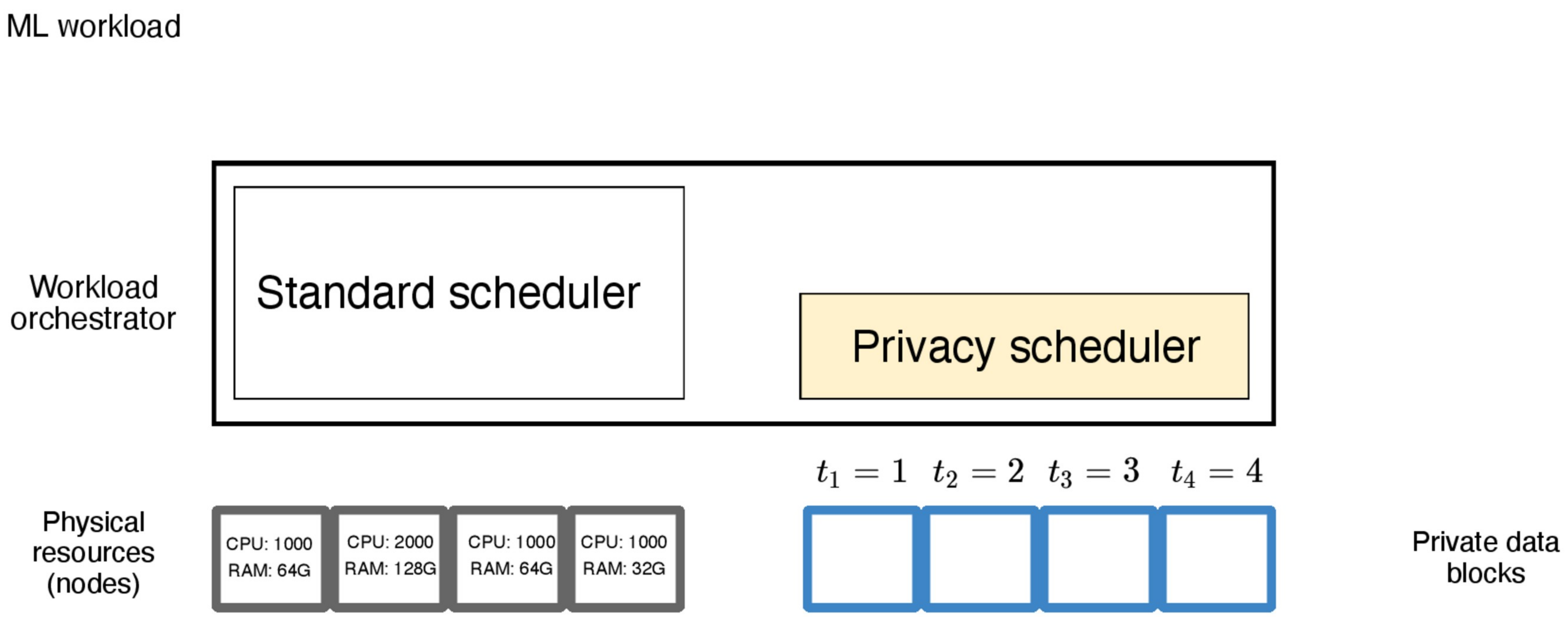
PrivateKube

- Extension for Kubernetes that adds privacy as a **new resource** alongside traditional compute resources
- New scheduler: **Dominant Privacy Fairness (DPF)**, a variant of Dominant Resource Fairness (DRF)
- DPF enjoys similar **fairness properties** as DRF, with some definitional changes to account for privacy characteristics

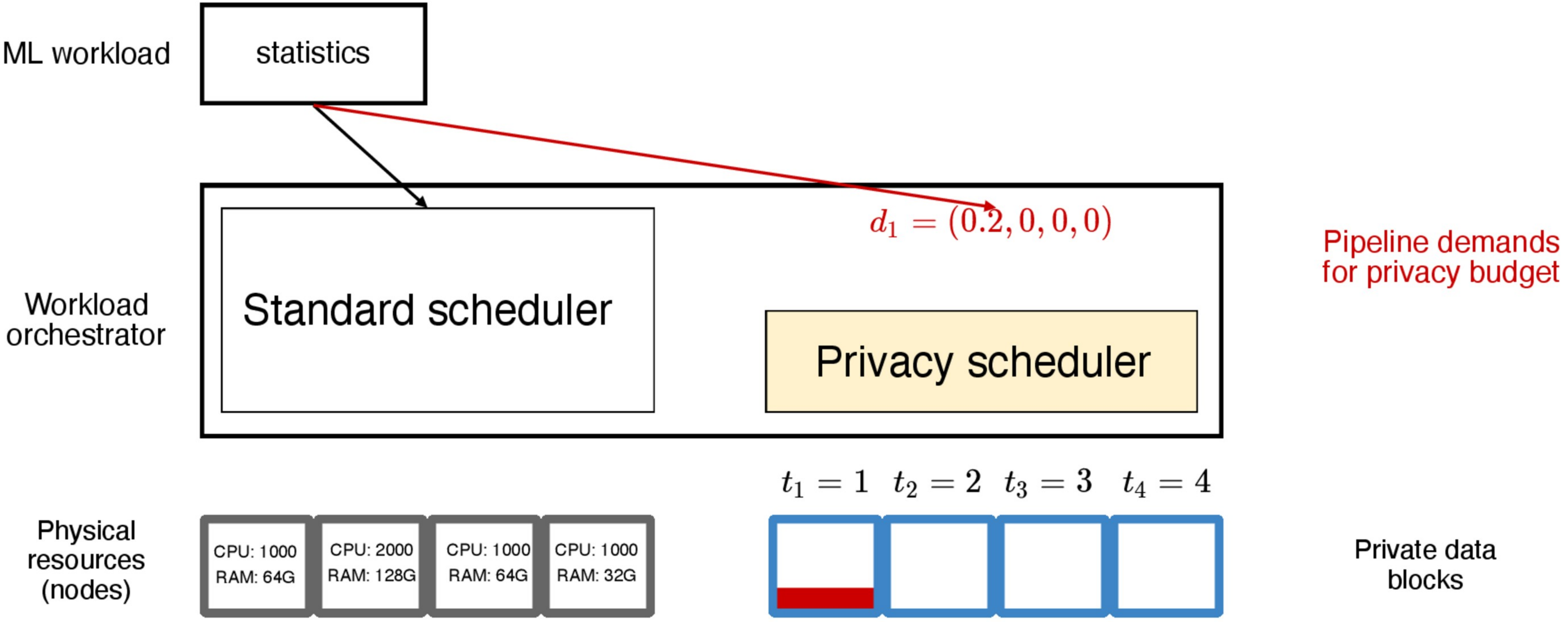
Outline

1. Motivation
2. **Architecture**
3. DPF scheduler
4. Evaluation

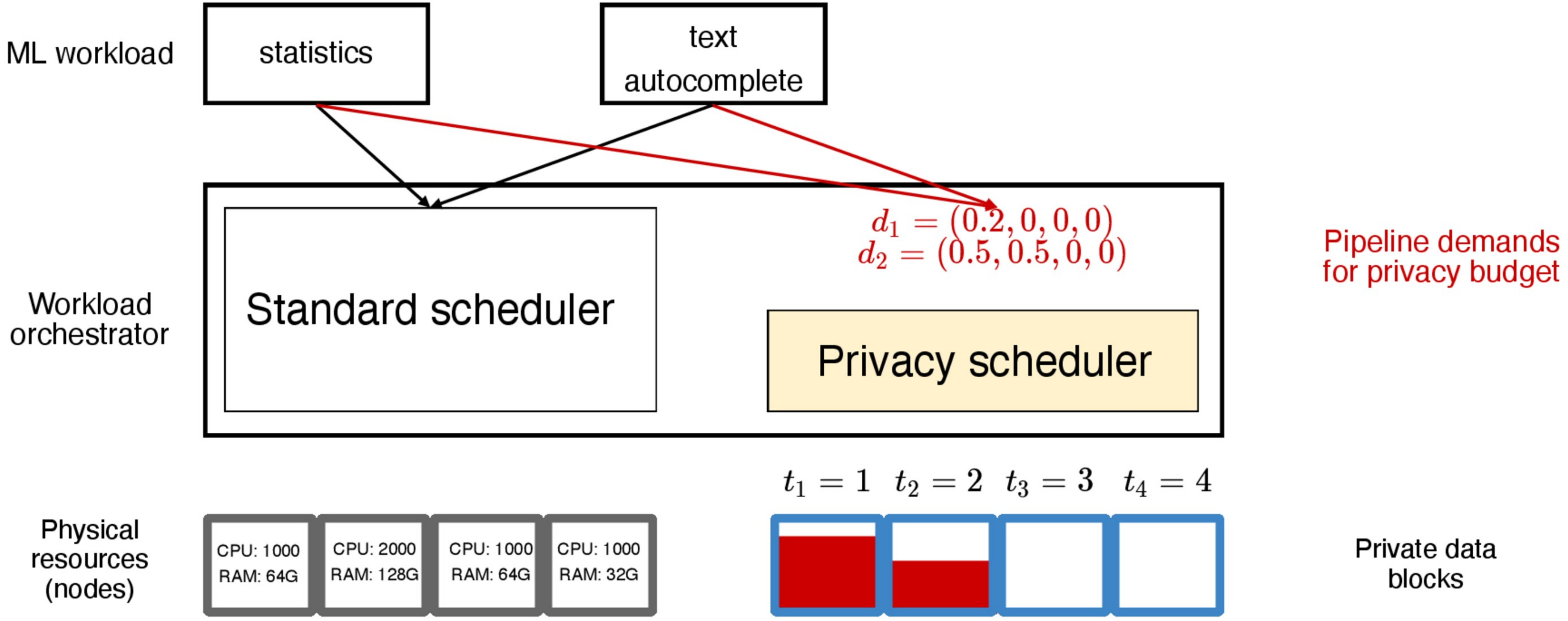
Architecture



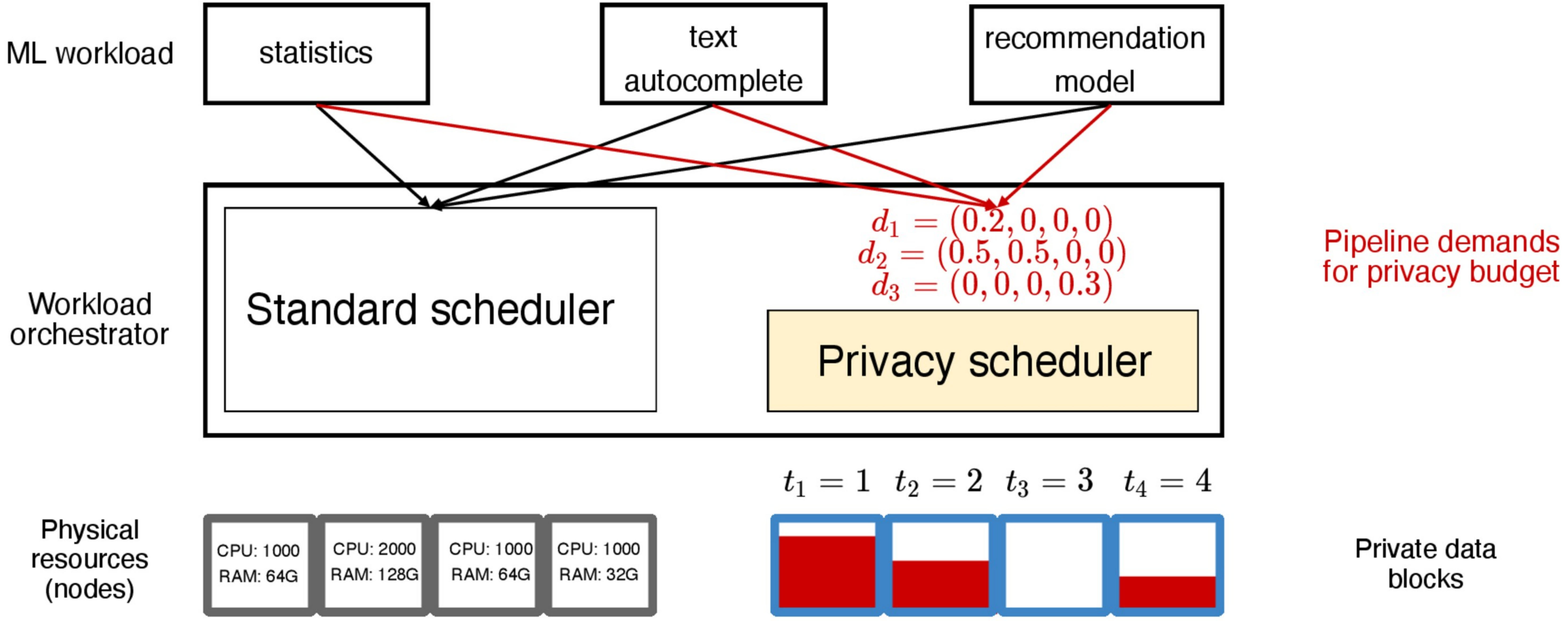
Architecture



Architecture



Architecture



Outline

1. Motivation
2. Architecture
3. **DPF scheduler**
4. Evaluation

DRF as a Basis

- Dominant Resource Fairness (DRF) to allocate **multiple resources** (Ghodsi+11)

- Popular for datacenters (CPU, GPU, RAM)

- For compute, it gives **max-min fairness** over m resources

Algorithm 1. DRF

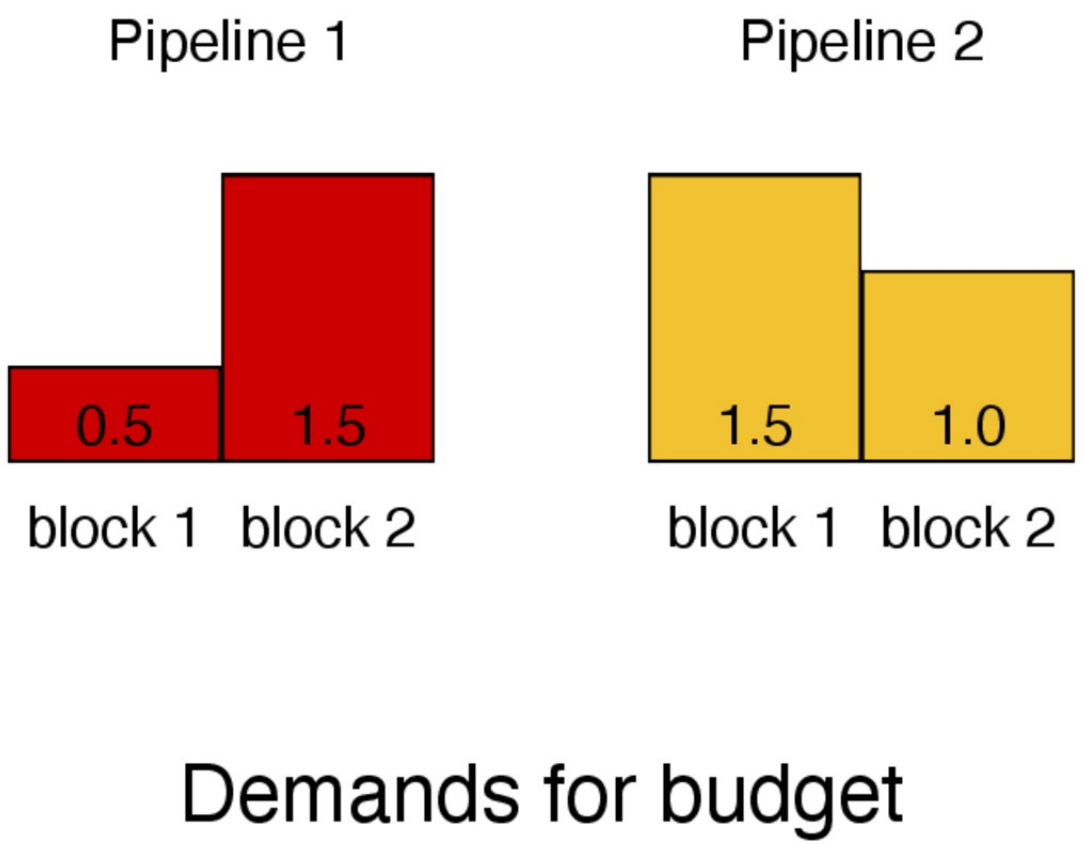
$R = \langle R_1, \ldots, R_m \rangle$ resource capacities
$C = \langle C_1, \ldots, C_m \rangle$ consumed resources

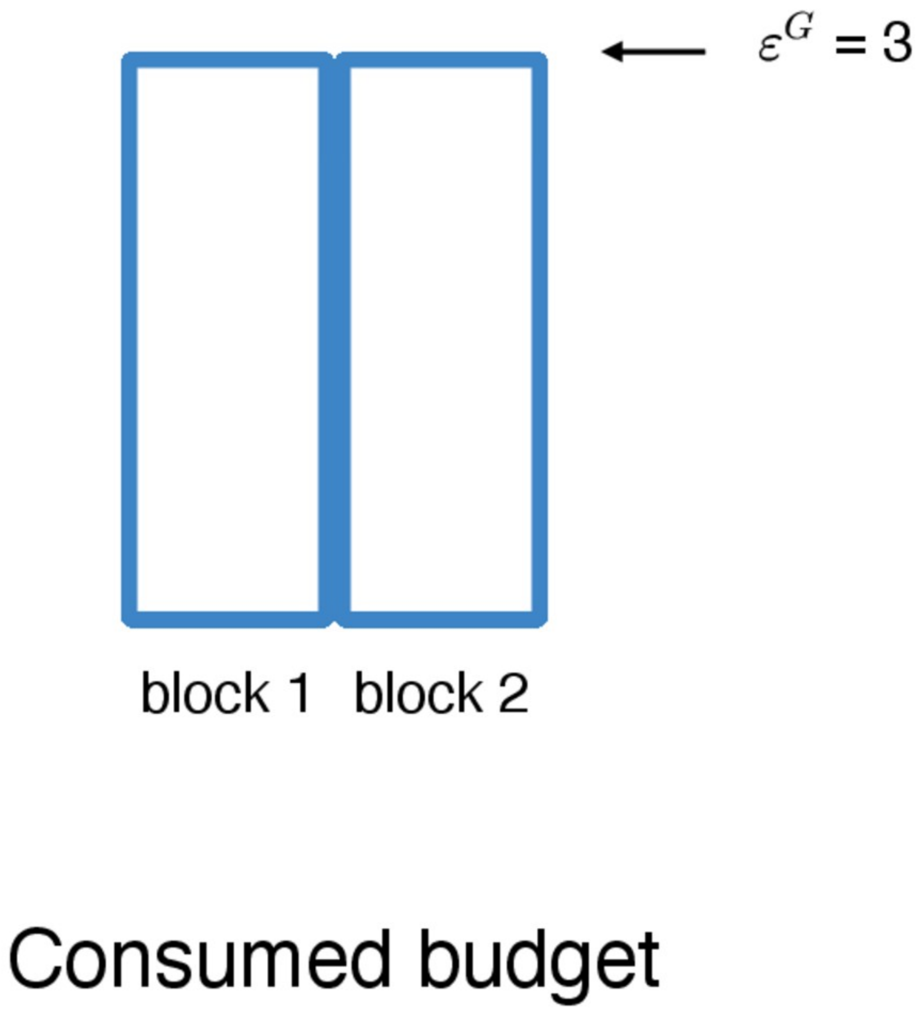
$\text{DominantShare}(d_i) := \max_j \frac{d_{i,j}}{R_j}$

$\text{OnSchedulerTimer}(WaitingJobs):$
$\quad SortedJobs \leftarrow \text{sortBy}(\text{DominantShare}, WaitingJobs)$
$\quad \text{for } i \in SortedJobs:$
$\quad\quad \text{if } C + d_i \leq R:$
$\quad\quad\quad C \leftarrow C + d_i$

Problem: Privacy is not Replenishable

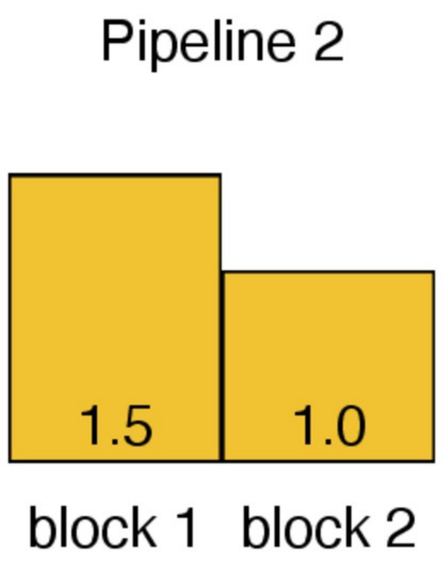
$$t = 1$$



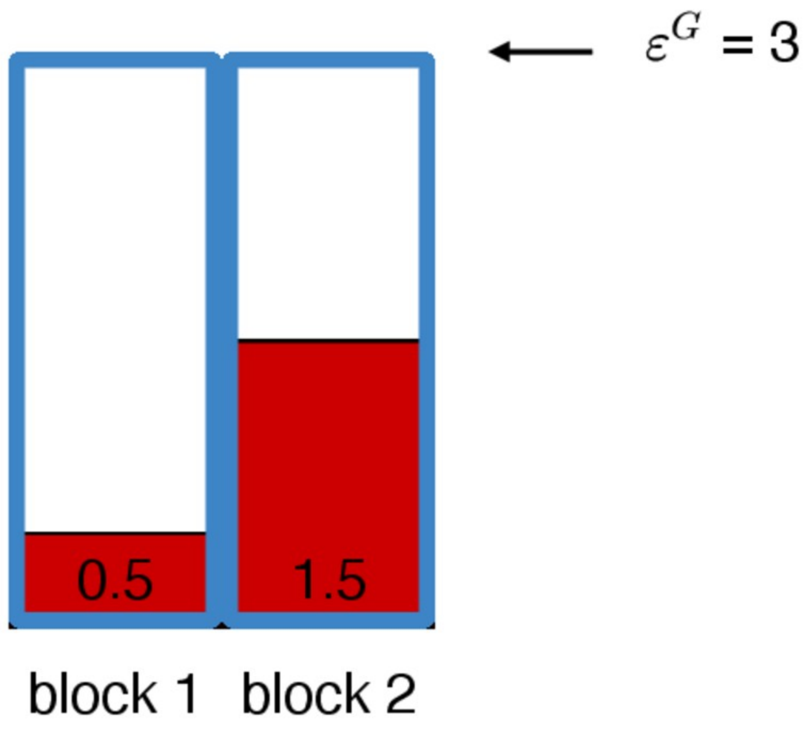


Problem: Privacy is not Replenishable

$$t = 1$$



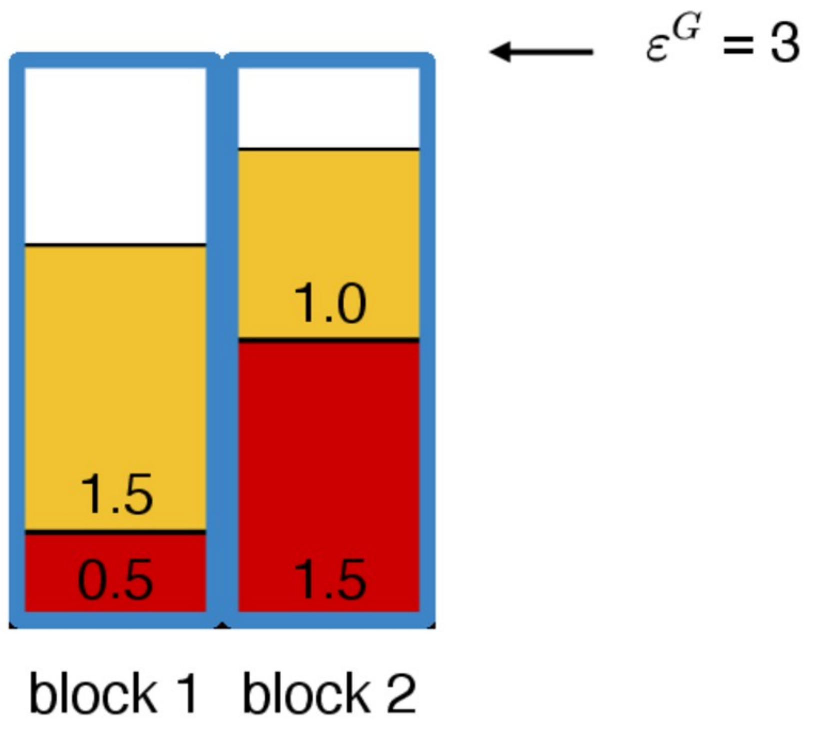
Demands for budget



Consumed budget

Problem: Privacy is not Replenishable

$t = 1$

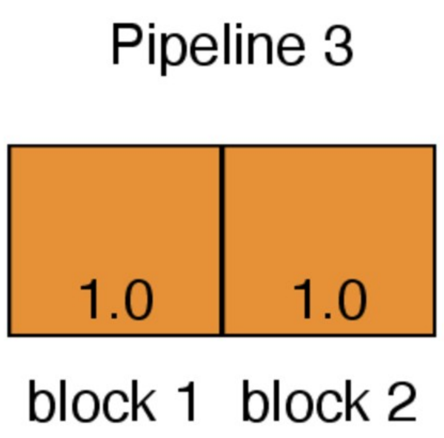


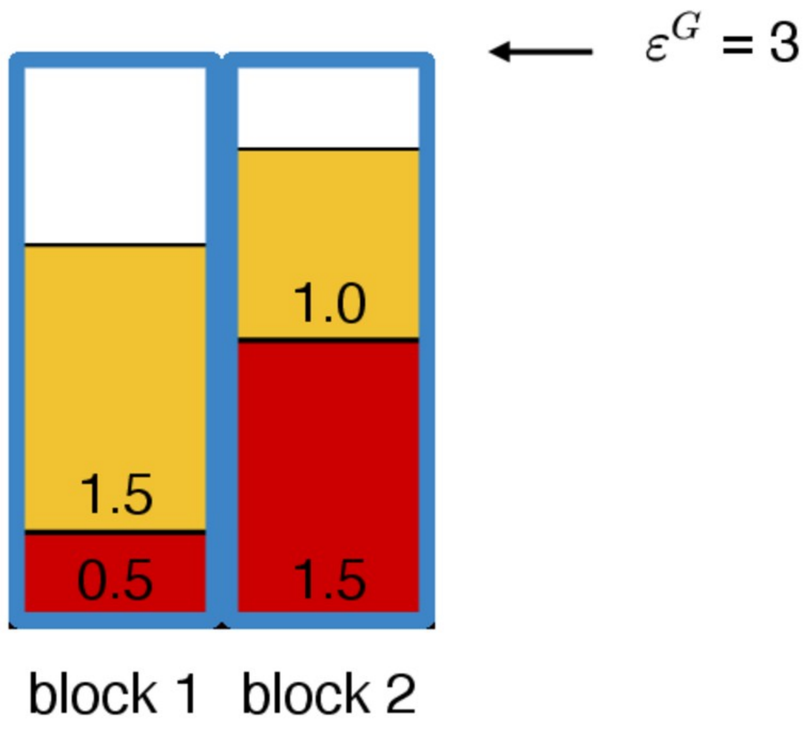
Demands for budget

Consumed budget

Problem: Privacy is not Replenishable

$$t = 2$$





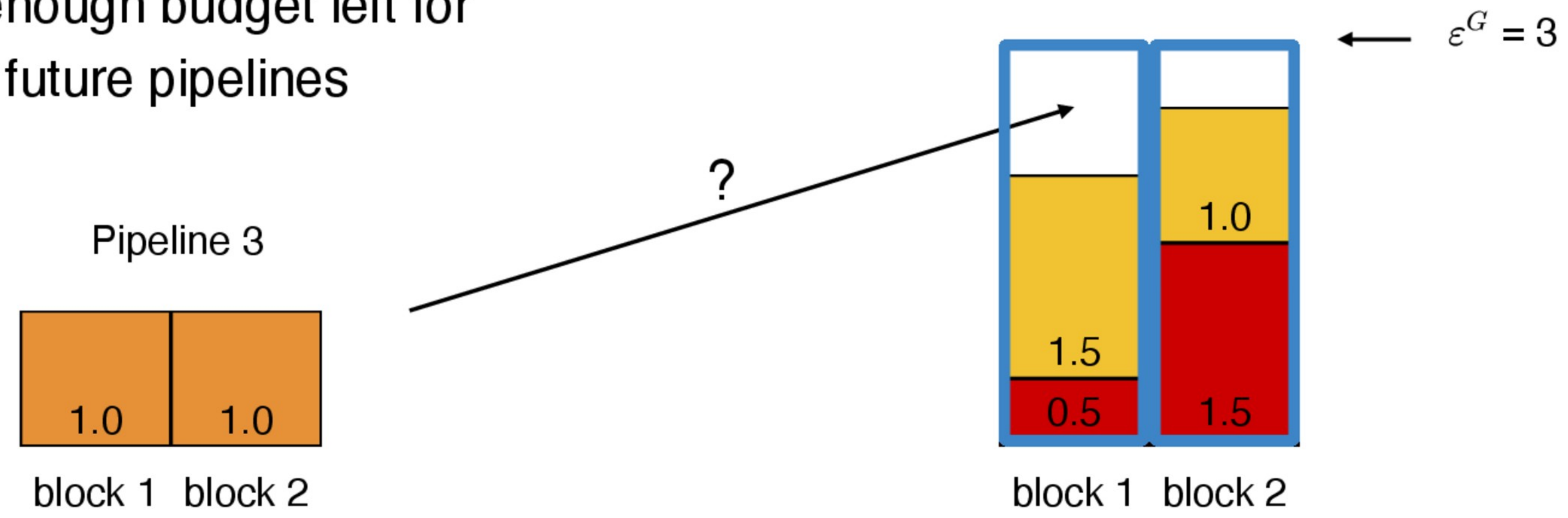
Demands for budget

Consumed budget

Problem: Privacy is not Replenishable

$t = 2$

Not enough budget left for future pipelines



Demands for budget

Consumed budget

Dominant Privacy Fairness (DPF)

- Idea: **unlock privacy budget** for each block progressively, so budget remains for the future

- Like DRF but only for the first N pipelines for each block, and best-effort scheduling for the others

Algorithm 2. DPF-N

$R = \langle R_1, \ldots, R_m \rangle$ private block capacities (aka ε^G)
$C = \langle C_1, \ldots, C_m \rangle$ consumed budgets
$U = \langle U_1, \ldots, U_m \rangle$ unlocked budgets (initially 0)

Dominant Privacy Fairness (DPF)

- Idea: **unlock privacy budget** for each block progressively, so budget remains for the future
- Like DRF but only for the first N pipelines for each block, and best-effort scheduling for the others

Algorithm 2. DPF-N

$R = \langle R_1, \ldots, R_m \rangle$ private block capacities (aka ε^G)
$C = \langle C_1, \ldots, C_m \rangle$ consumed budgets
$U = \langle U_1, \ldots, U_m \rangle$ unlocked budgets (initially 0)

$\mathrm{OnPipelineArrival}(d_i):$
  for $j \in \{j : d_{i,j} > 0\}:$
    $U_j \leftarrow \min(R_j, U_j + \frac{R_j}{N})$

Dominant Privacy Fairness (DPF)

- Idea: **unlock privacy budget** for each block progressively, so budget remains for the future
- Like DRF but only for the first N pipelines for each block, and best-effort scheduling for the others

Algorithm 2. DPF-N

$R = \langle R_1, \ldots, R_m \rangle$ private block capacities (aka ε^G)
$C = \langle C_1, \ldots, C_m \rangle$ consumed budgets
$U = \langle U_1, \ldots, U_m \rangle$ unlocked budgets (initially 0)

$\text{OnPipelineArrival}(d_i)$:
  for $j \in \{j : d_{i,j} > 0\}$:
    $U_j \leftarrow \min(R_j, U_j + \frac{R_j}{N})$

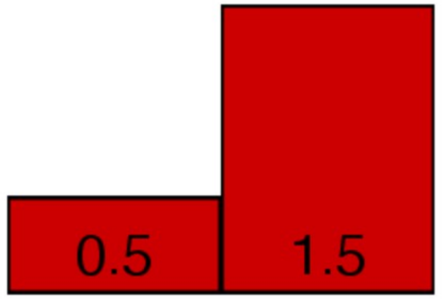
$\text{DominantShare}(d_i) := \max_j \frac{d_{i,j}}{R_j}$

$\text{OnSchedulerTimer}(WaitingJobs)$:
  $SortedJobs \leftarrow \text{sortBy}(\text{DominantShare}, WaitingJobs)$
  for $i \in SortedJobs$:
    if $C + d_i \leq U$:
      $C \leftarrow C + d_i$

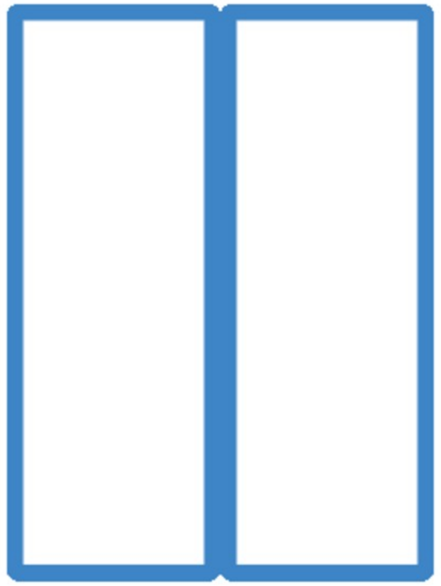
DPF Example

$t = 1$

Pipeline 1



block 1 block 2



block 1 block 2

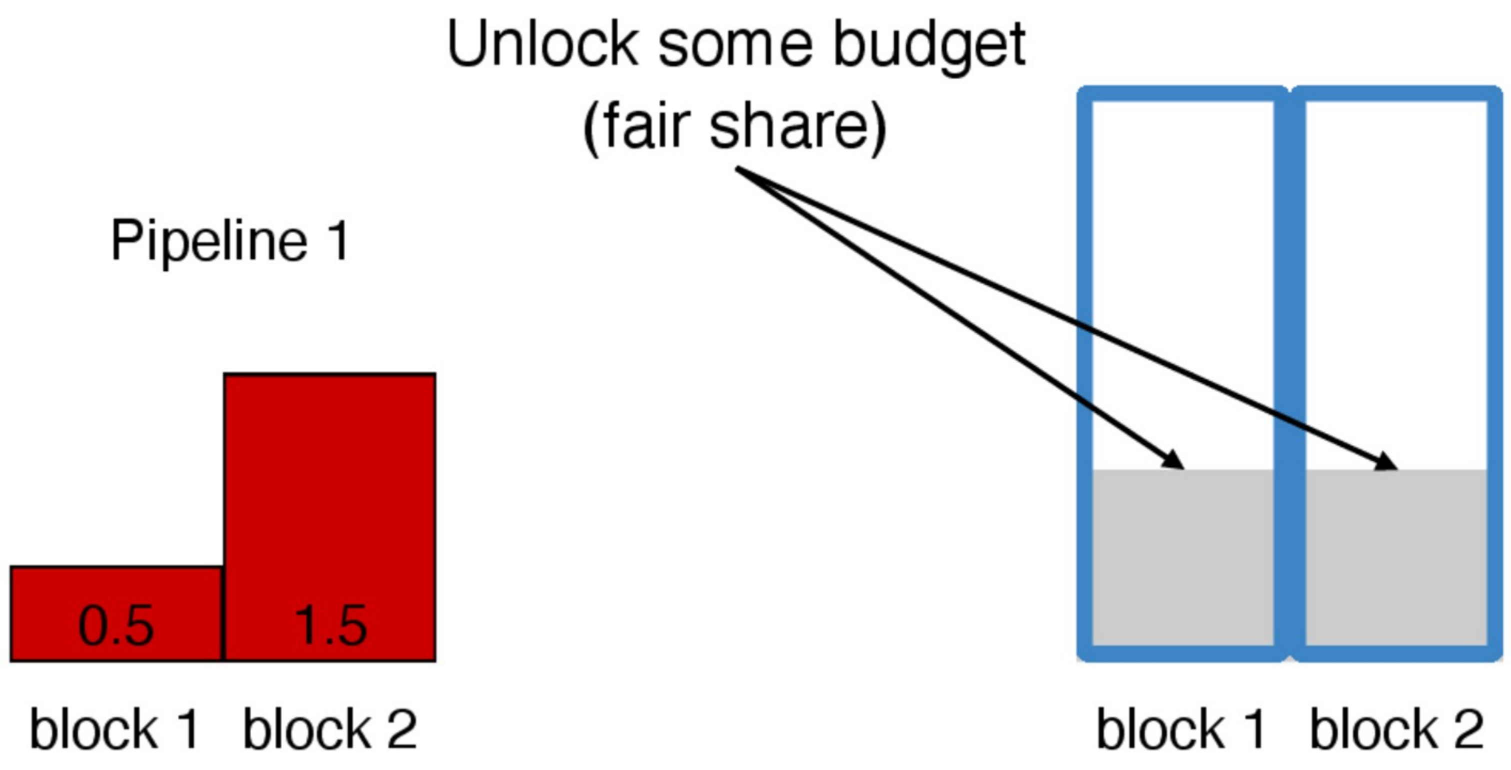
Incoming pipelines

DPF queue

Allocated budget

DPF Example

$$t = 1$$



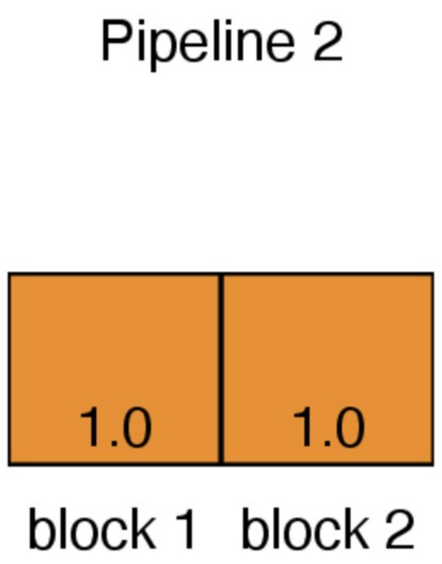
Incoming pipelines

DPF queue

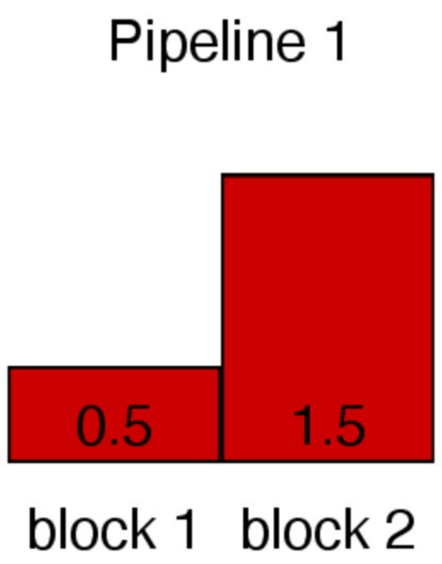
Allocated budget

DPF Example

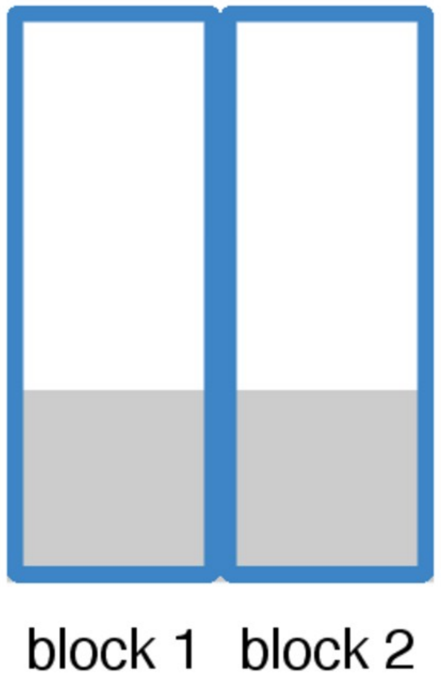
$$t = 2$$



Incoming pipelines



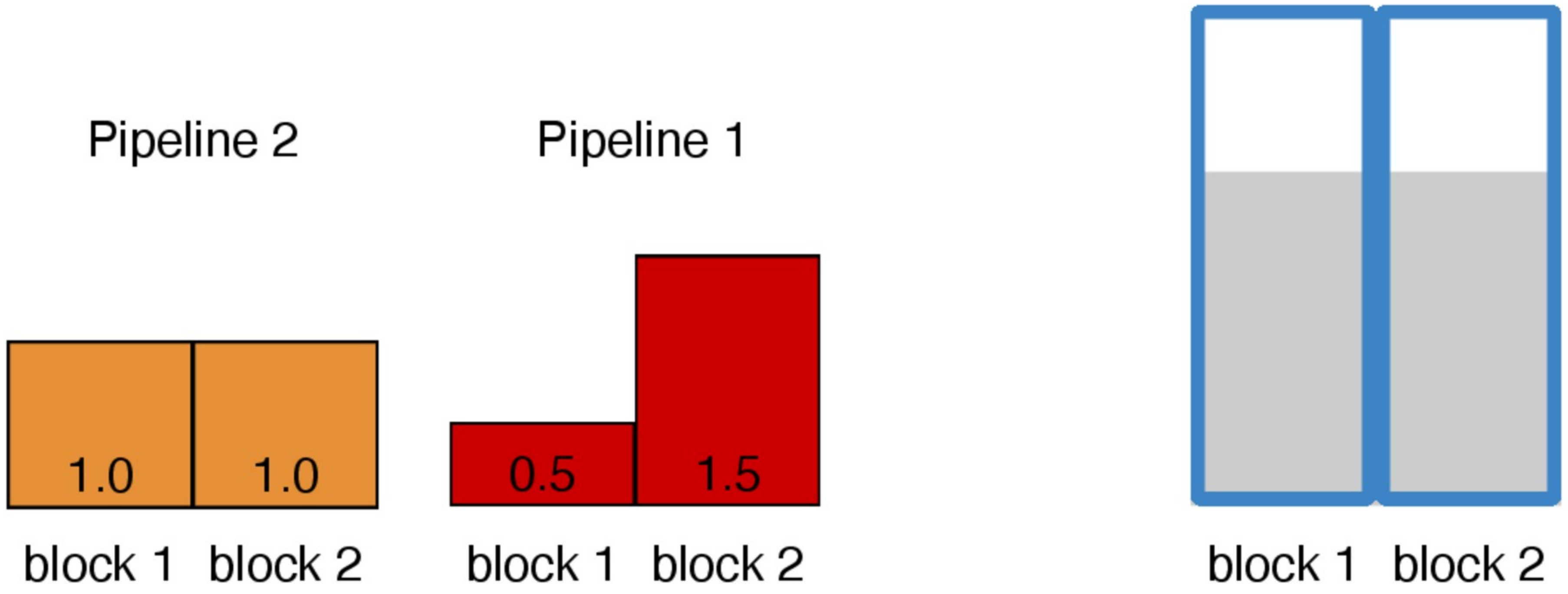
DPF queue



Allocated budget

DPF Example

$$t = 2$$

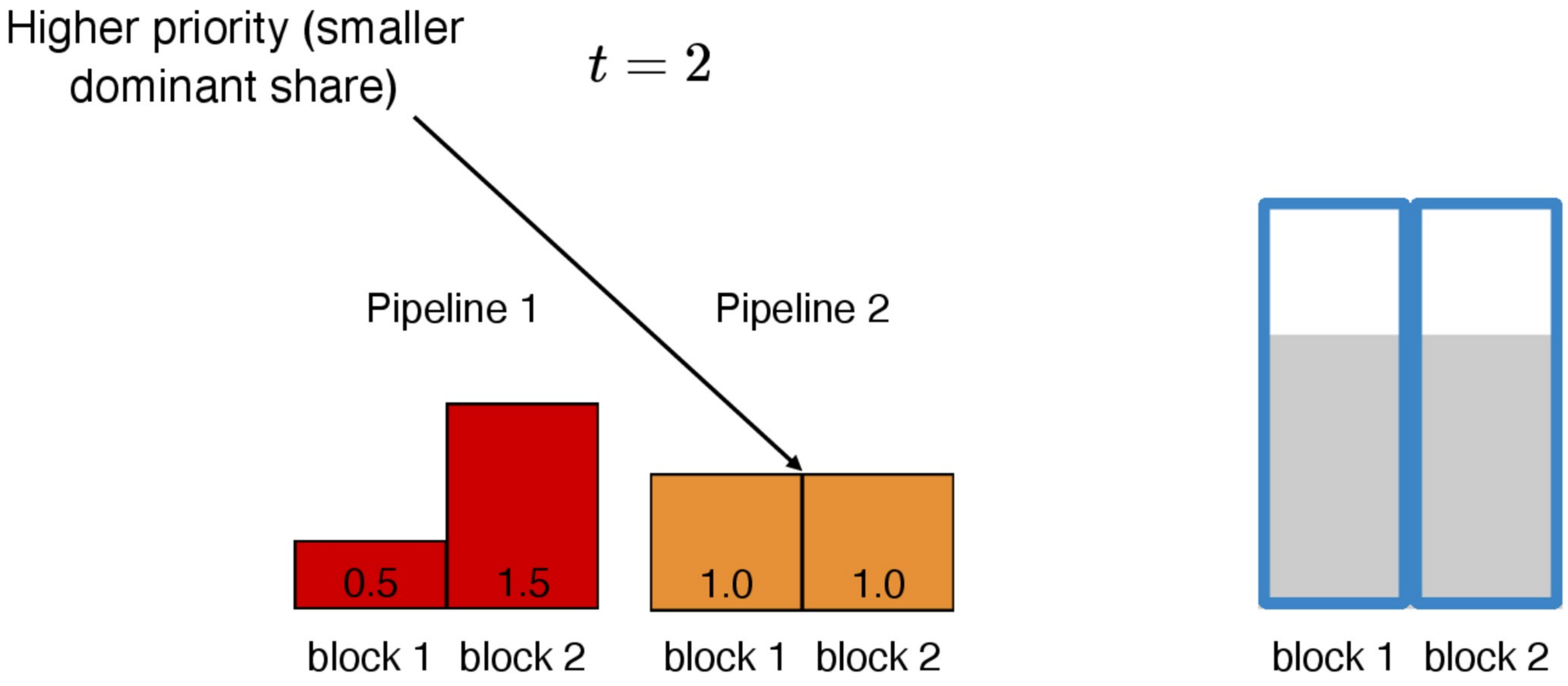


Incoming pipelines

DPF queue

Allocated budget

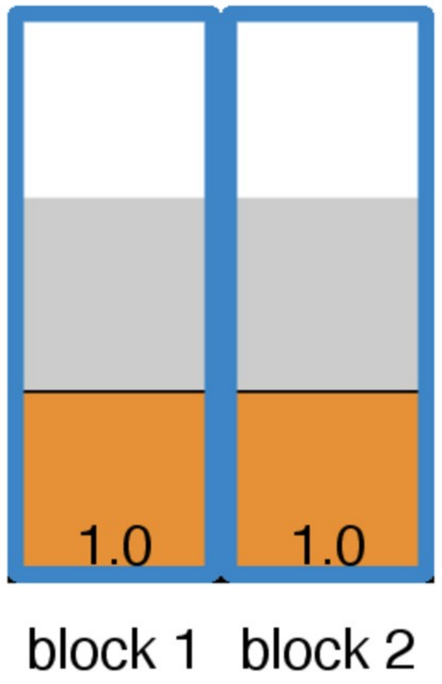
DPF Example



Incoming pipelines DPF queue Allocated budget

DPF Example

$$t = 2$$



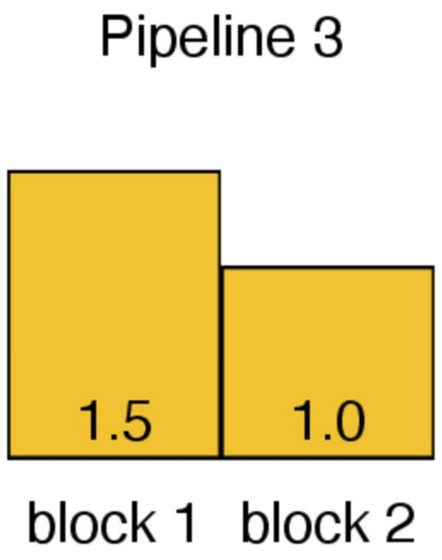
Incoming pipelines

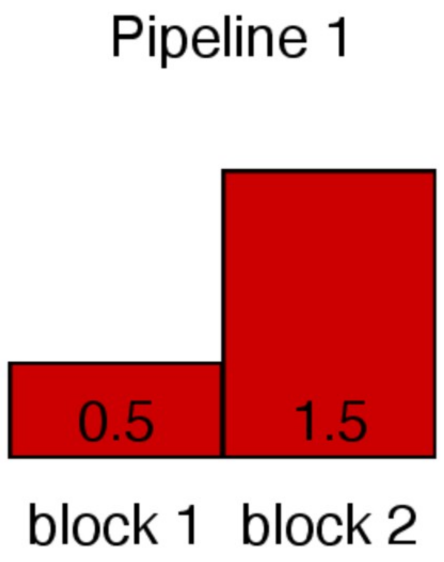
DPF queue

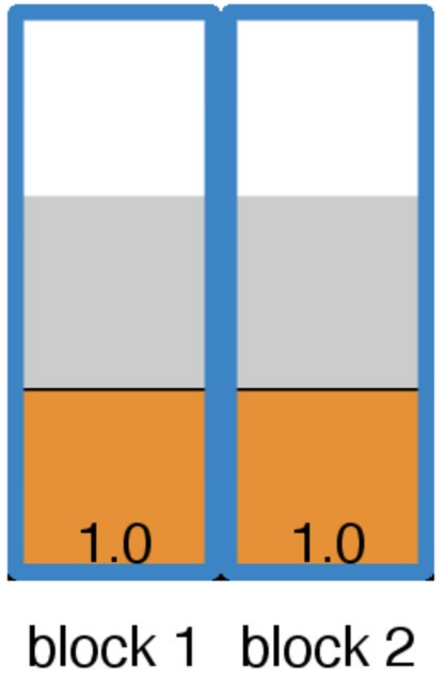
Allocated budget

DPF Example

$$t = 3$$







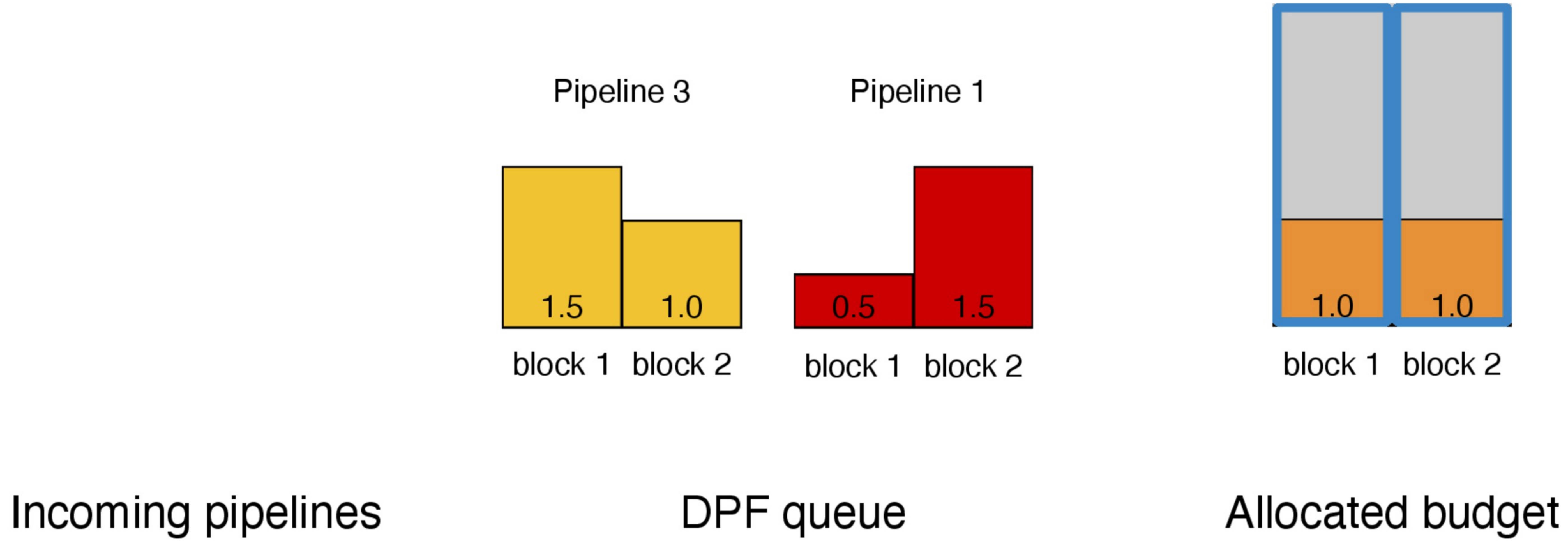
Incoming pipelines

DPF queue

Allocated budget

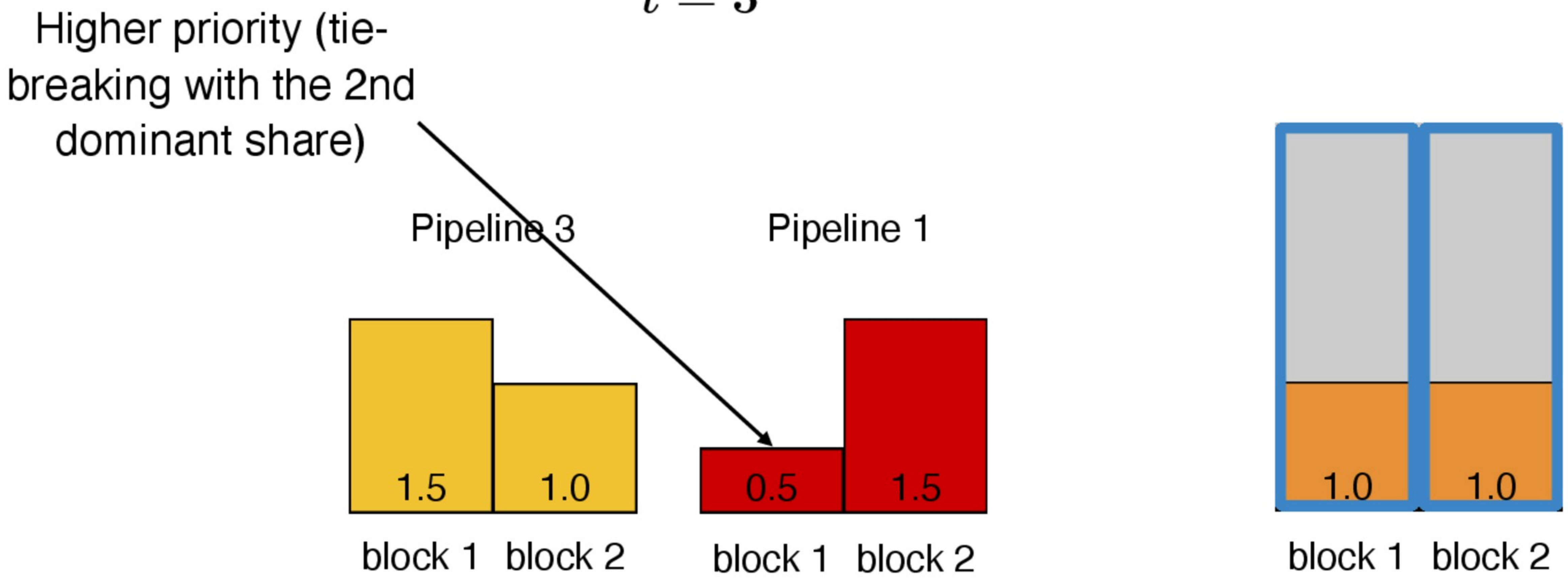
DPF Example

$$t = 3$$



DPF Example

$t = 3$



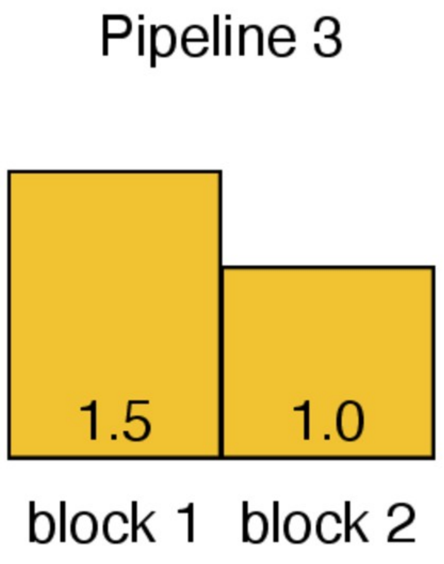
Incoming pipelines

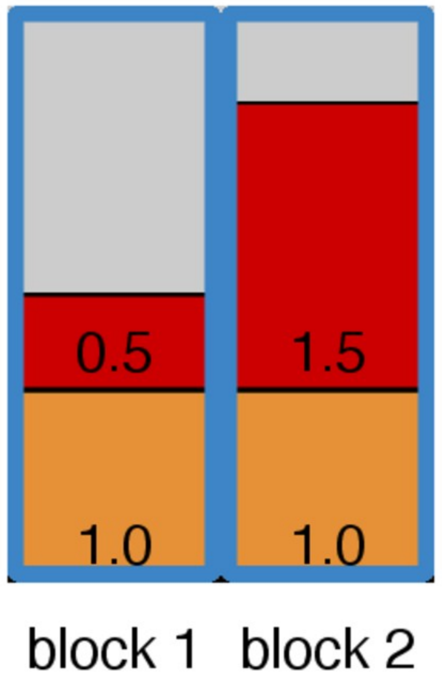
DPF queue

Allocated budget

DPF Example

$$t = 3$$





Incoming pipelines

DPF queue

Allocated budget

DPF Properties

Max-min fairness **only for the first N pipelines** over any block

Game theoretic properties:

- sharing incentive
- strategy-proofness
- dynamic envy-freeness
- Pareto-efficiency

Definition. A pipeline is a *fair demand pipeline* if:

a) its demand for each one of the blocks is smaller than the fair share ϵ_j^G/N and

b) it is within the first N pipelines that requested some budget for all its requested blocks

Outline

1. Motivation
2. Architecture
3. DPF scheduler
4. **Evaluation**

Methodology

Questions

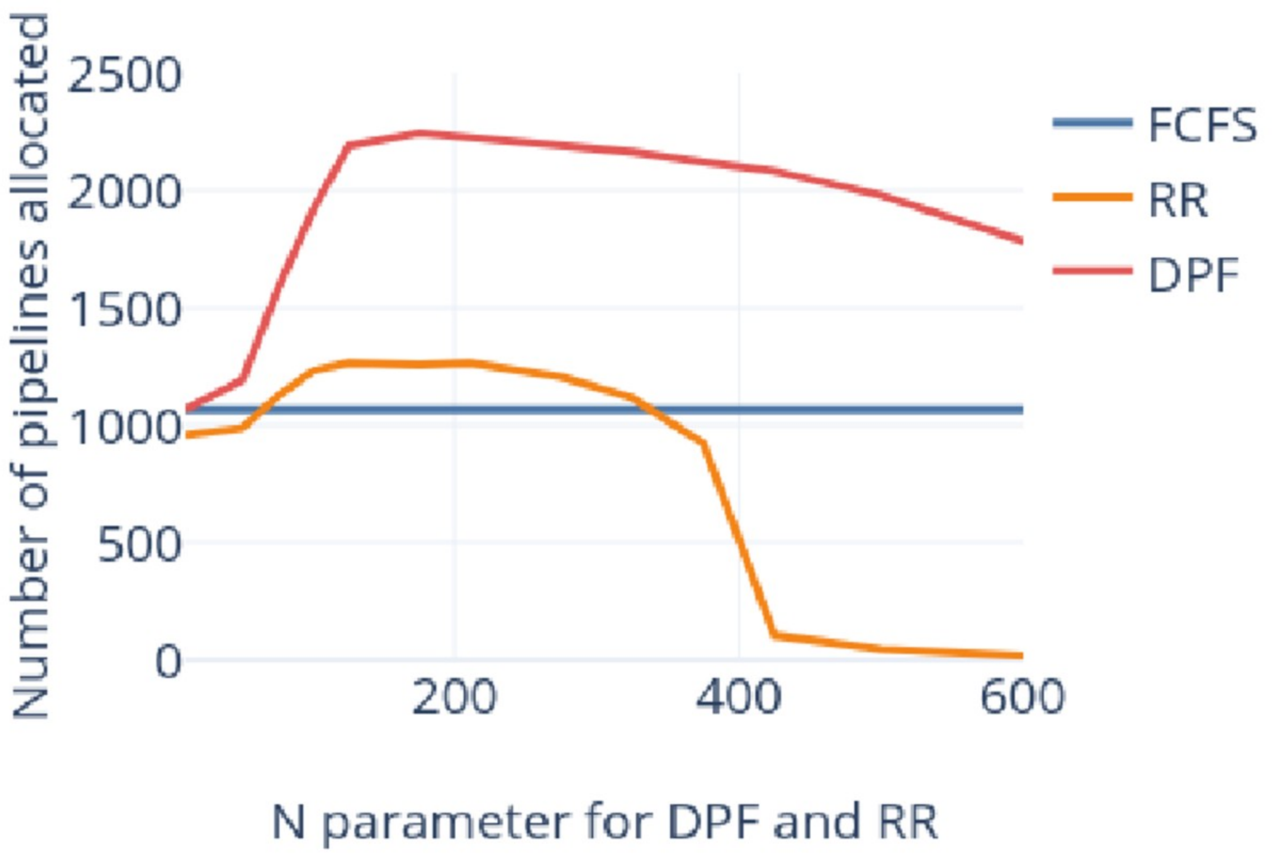
- **How does DPF compare to baseline schedulers?**
- How do workload characteristics impact DPF?
- How does the DP semantic impact DPF?

Workloads

- **Microbenchmark**: $\varepsilon \in \{0.01\varepsilon^G, 0.1\varepsilon^G\}$, either the last block or the 10 last blocks
- Macrobenchmark: NLP pipelines and summary statistics over the Amazon Reviews dataset with various demands

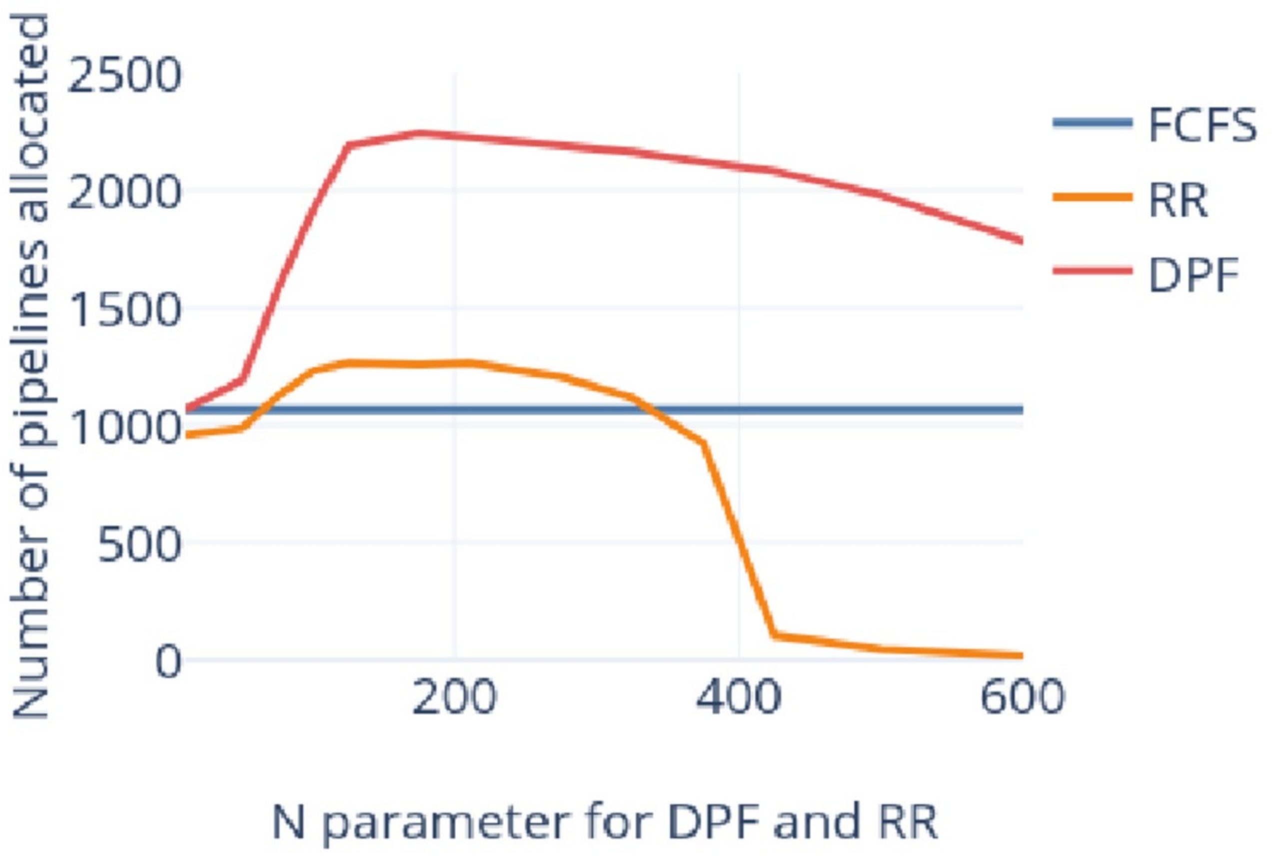
How does DPF compare to baseline schedulers?

Allocation

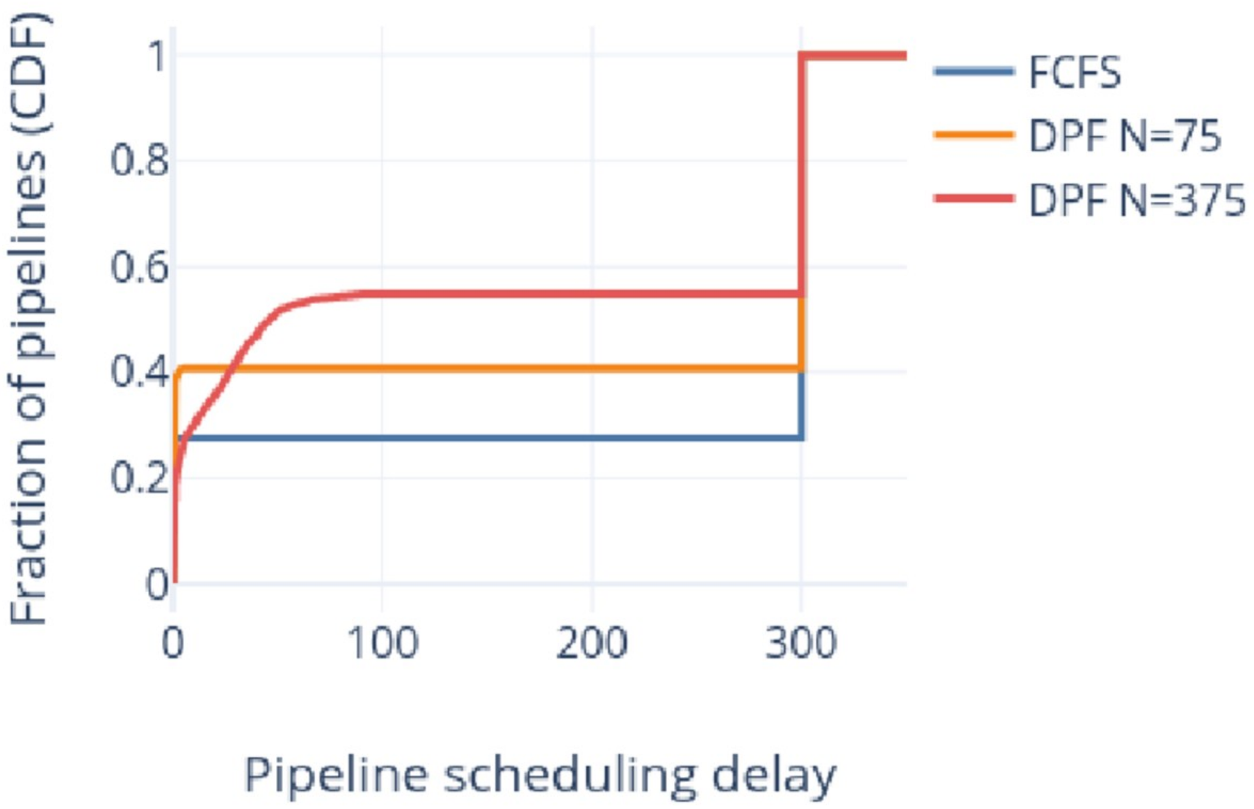


How does DPF compare to baseline schedulers?

Allocation

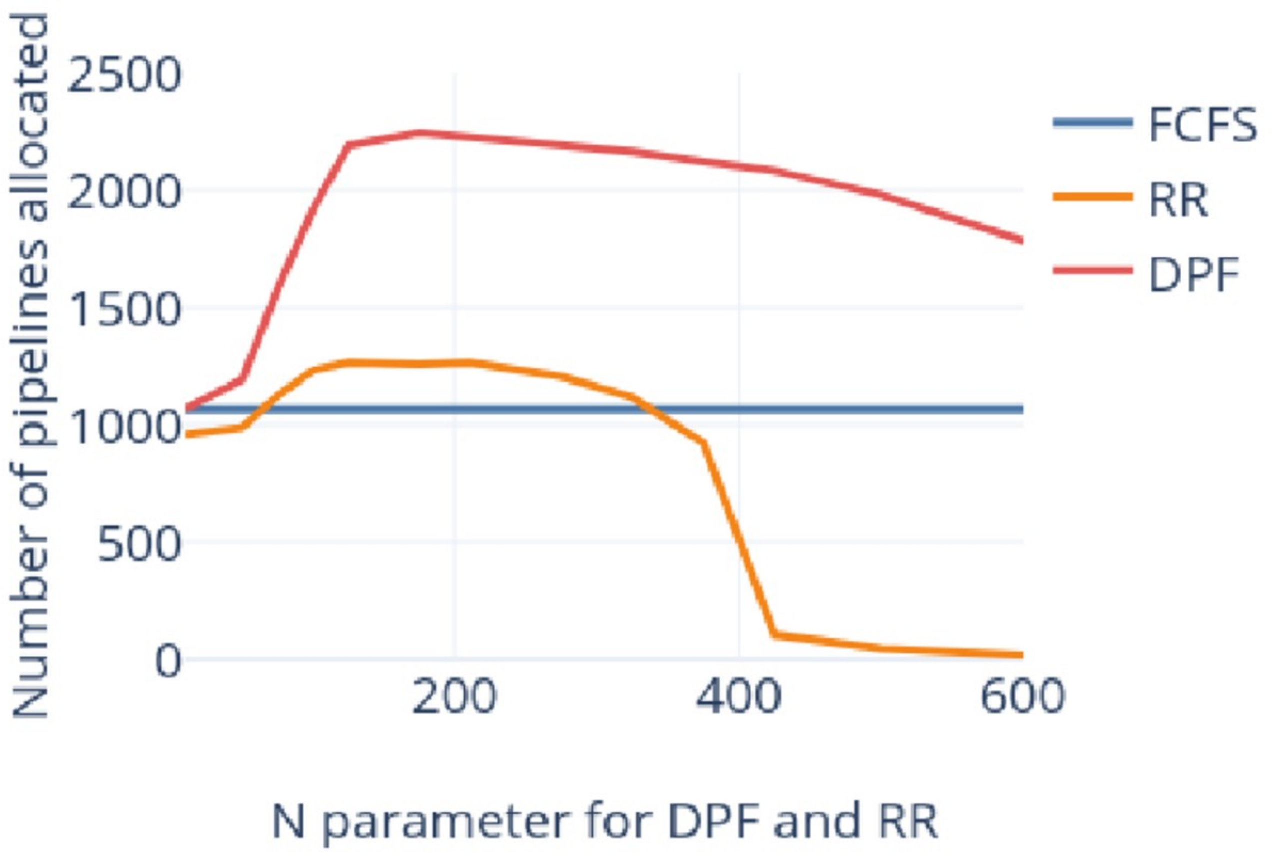


Latency

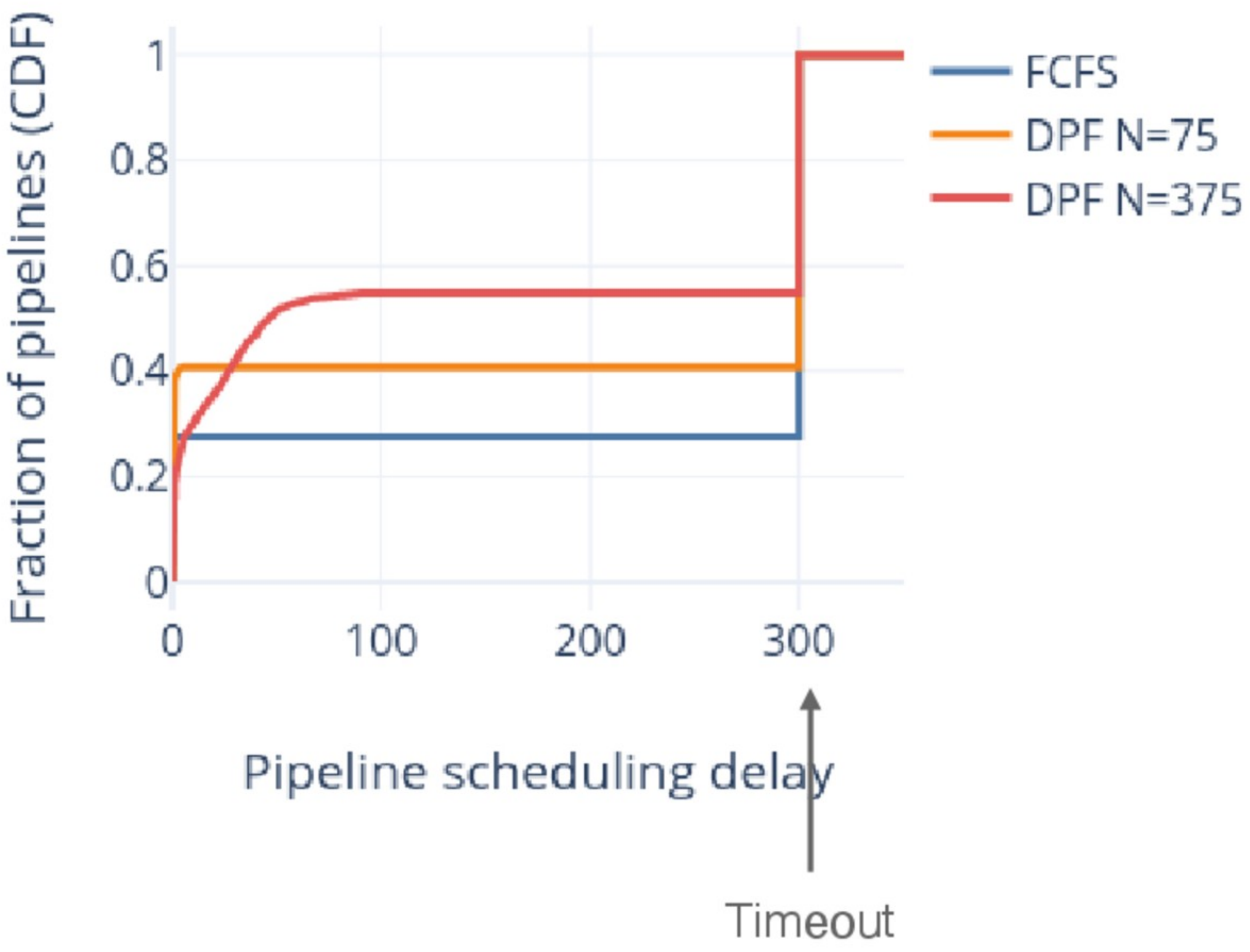


How does DPF compare to baseline schedulers?

Allocation



Latency



Conclusion

- **Privacy as a resource** that should be tracked and scheduled

- PrivateKube incorporates privacy as a new resource into Kubernetes and provides **Dominant Privacy Fairness (DPF)**, the first scheduling algorithm suitable for this non-replenishable resource.

- Changes to the algorithm and fairness definitions show that **scheduling privacy is a new problem**, for which more work is needed.

Code and paper: https://columbia.github.io/PrivateKube

Contact: pierre@cs.columbia.edu

References

(Carlini+18) Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, Colin Raffel. *Extracting Training Data from Large Language Models*. arxiv 2020.

(Shokri+17) Reza Shokri, Marco Stronati, Congzheng Song, Vitaly Shmatikov. *Membership Inference Attacks against Machine Learning Models*. S&P 2017.

(Calandrino+11) J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov. *"You Might Also Like:" Privacy risks of collaborative filtering*. S&P 2011.

(Lecuyer+19) M. Lecuyer, R. Spahn, R. Geambasu, and D. Hsu. *Privacy Accounting and Quality Control in the Sage Differentially Private Machine Learning Platform*. SOSP 2019.

(Ghodsi+11) Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, Ion Stoica. *Dominant Resource Fairness: Fair Allocation of Multiple Resource Types*. NSDI 2011.

References

(Zanella-Béguelin+20) Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, Marc Brockschmidt. *Analyzing Information Leakage of Updates to Natural Language Models*. CCS 2020.

(Dinur+Nissim-03) I. Dinur and K. Nissim. *Revealing information while preserving privacy*. PODS, 2003.