Privacy Budget Scheduling

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

*Columbia University †Microsoft Research
Example: Messaging App

users, devices

functionality

database of user data

traditional code

Access control
Example: Messaging App

- Users, devices
- Models/predictions
- Functionality
- Database of user data

- ML model: autocomplete
- ML model: recommendation
- ML model: ad targeting

Access control

Traditional code
What Can Leak?

privacy attacks

functionality

database of user data

SSN, address, ... (Carlini+20)

user actions (Calandrino+11)

membership (Shokri+17)

ML model: autocomplete

ML model: recommendation

ML model: ad targeting

traditional code
Access control
Is Differential Privacy (DP) the Solution?

privacy attacks

functionality

database of user data

ML model: autocomplete

ML model: recommendation

ML model: ad targeting

traditional code

Access control

DP

DP

DP
Is Differential Privacy (DP) the Solution?

Yes, but it depends at which level we apply it

privacy attacks

functionality

database of user data

ML model: autocomplete
ML model: recommendation
ML model: ad targeting

traditional code
Access control
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 $(\varepsilon, \delta)$-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

ML model: autocomplete
\( \varepsilon_1 \) DP

ML model: recommendation
\( \varepsilon_2 \) DP

ML model: ad targeting
\( \varepsilon_3 \) DP
Problem: Privacy Loss Accumulates

ML model: autocomplete
\(\varepsilon_1\) DP

ML model: recommendation
\(\varepsilon_2\) DP

ML model: ad targeting
\(\varepsilon_3\) DP
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?

- ML model: autocomplete
  - $\varepsilon_1$ DP

- ML model: recommendation
  - $\varepsilon_2$ DP

- ML model: ad targeting
  - $\varepsilon_3$ DP

Workload of multiple, repeatedly trained models

Growing database of user data
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** $\epsilon_G$

  \[
  \sum_{\text{task } i} \epsilon_i \leq \epsilon_G.
  \]

- 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

ML workload

Workload orchestrator

Standard scheduler

Privacy scheduler

Physical resources (nodes)

CPU: 1000
RAM: 64G

CPU: 2000
RAM: 128G

CPU: 1000
RAM: 64G

CPU: 1000
RAM: 32G

$t_1 = 1 \quad t_2 = 2 \quad t_3 = 3 \quad t_4 = 4$

Private data blocks
Architecture

ML workload

statistics

Standard scheduler

Pipeline demands for privacy budget

d_1 = (0.2, 0, 0, 0)

Privacy scheduler

Physical resources (nodes)

CPU: 1000
RAM: 64G

CPU: 2000
RAM: 128G

CPU: 1000
RAM: 64G

CPU: 1000
RAM: 32G

Private data blocks

\[ t_1 = 1 \quad t_2 = 2 \quad t_3 = 3 \quad t_4 = 4 \]
Architecture

ML workload

- statistics
- text autocomplete

Standard scheduler

Privacy scheduler

Physical resources (nodes)
- CPU: 1000
  - RAM: 64G
- CPU: 2000
  - RAM: 128G
- CPU: 1000
  - RAM: 64G
- CPU: 1000
  - RAM: 32G

Pipeline demands for privacy budget

$\begin{align*}
  d_1 &= (0.2, 0, 0, 0) \\
  d_2 &= (0.5, 0.5, 0, 0)
\end{align*}$

Private data blocks

$t_1 = 1 \quad t_2 = 2 \quad t_3 = 3 \quad t_4 = 4$
Architecture

ML workload

statistics

text autocomplete

recommendation model

Standard scheduler

Privacy scheduler

Physical resources (nodes)

CPU: 1000
RAM: 64G

CPU: 2000
RAM: 128G

CPU: 1000
RAM: 64G

CPU: 1000
RAM: 32G

Pipeline demands for privacy budget

d_1 = (0.2, 0.0, 0.0)
d_2 = (0.5, 0.5, 0.0)
d_3 = (0.0, 0.0, 0.3)

\[ t_1 = 1 \quad t_2 = 2 \quad t_3 = 3 \quad t_4 = 4 \]

Private data blocks
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

1. \( R = \langle R_1, \ldots, R_m \rangle \) resource capacities
2. \( C = \langle C_1, \ldots, C_m \rangle \) consumed resources

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

OnSchedulerTimer(\text{WaitingJobs}):

1. \( \text{SortedJobs} \leftarrow \text{sortBy} (\text{DominantShare}, \text{WaitingJobs}) \)
2. for \( i \in \text{SortedJobs} : \)
   1. if \( C + d_i \leq R : \)
      1. \( C \leftarrow C + d_i \)
Problem: Privacy is not Replenishable

\[ t = 1 \]

Demands for budget

Consumed budget

\[ \varepsilon^G = 3 \]
Problem: Privacy is not Replenishable

\[ t = 1 \]

Demands for budget

Consumed budget

\[ \epsilon^G = 3 \]
Problem: Privacy is not Replenishable

$t = 1$

Demands for budget

Consumed budget

$\epsilon^G = 3$
Problem: Privacy is not Replenishable

\[ t = 2 \]

Demands for budget

Consumed budget

\[ \epsilon^G = 3 \]
Problem: Privacy is not Replenishable

\[ t = 2 \]

Not enough budget left for future pipelines

Demands for budget

Consumed budget

\[ \varepsilon^G = 3 \]
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 $\varepsilon^G$)
$C = \langle C_1, \ldots, C_m \rangle$ 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 $\varepsilon^G$)
$C = \langle C_1, \ldots, C_m \rangle$ consumed budgets
$U = \langle U_1, \ldots, U_m \rangle$ unlocked budgets (initially 0)

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 $\varepsilon^G$)

$$C = \langle C_1, \ldots, C_m \rangle$$ consumed budgets

$$U = \langle U_1, \ldots, U_m \rangle$$ unlocked budgets (initially 0)

\[
\text{OnPipelineArrival}(d_i) :
\]

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

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

\[
\text{OnSchedulerTimer(WaitingJobs)} :
\]

\[
\text{SortedJobs} \leftarrow \text{sortBy}(\text{DominantShare}, \text{WaitingJobs})
\]

\[
\text{for } i \in \text{SortedJobs} : \\
\text{if } C + d_i \leq U : \\
C \leftarrow C + d_i
\]
DPF Example

$t = 1$

Incoming pipelines  DPF queue  Allocated budget
DPF Example

$t = 1$

Unlock some budget (fair share)

Incoming pipelines | DPF queue | Allocated budget

Pipeline 1

0.5 1.5

block 1  block 2

block 1  block 2
DPF Example

\[ t = 2 \]

Incoming pipelines

Pipeline 2

1.0 1.0

block 1 block 2

Pipeline 1

0.5 1.5

block 1 block 2

Allocated budget
DPF Example

\[ t = 2 \]

Incoming pipelines  |  DPF queue  |  Allocated budget

Pipeline 2
- block 1: 1.0
- block 2: 1.0

Pipeline 1
- block 1: 0.5
- block 2: 1.5

block 1 | block 2
DPF Example

Higher priority (smaller dominant share) $t = 2$

Pipeline 1
- block 1: 0.5
- block 2: 1.5

Pipeline 2
- block 1: 1.0
- block 2: 1.0

Incoming pipelines

DPF queue

Allocated budget
DPF Example

\[ t = 2 \]

Incoming pipelines

Pipeline 1

<table>
<thead>
<tr>
<th>block 1</th>
<th>block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>

DPF queue

<table>
<thead>
<tr>
<th>block 1</th>
<th>block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Allocated budget

<table>
<thead>
<tr>
<th>block 1</th>
<th>block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
DPF Example

$t = 3$

Incoming pipelines

Pipeline 3

1.5
block 1

1.0
block 2

Pipeline 1

0.5
block 1

1.5
block 2

Allocated budget

1.0
block 1

1.0
block 2
DPF Example

$t = 3$

Incoming pipelines

<table>
<thead>
<tr>
<th>Pipeline 3</th>
<th>Pipeline 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>0.5</td>
</tr>
<tr>
<td>block 1</td>
<td>block 1</td>
</tr>
<tr>
<td>1.0</td>
<td>1.5</td>
</tr>
<tr>
<td>block 2</td>
<td>block 2</td>
</tr>
</tbody>
</table>

DPF queue

<table>
<thead>
<tr>
<th>Allocated budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
</tr>
<tr>
<td>1.0</td>
</tr>
</tbody>
</table>
DPF Example

Higher priority (tie-breaking with the 2nd dominant share)

$t = 3$

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 \(e_j^G / N\) 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: $\epsilon \in \{0.01\epsilon^G, 0.1\epsilon^G\}$, 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**

![Graph showing allocation and latency comparisons between FCFS, RR, and DPF](image-url)
How does DPF compare to baseline schedulers?

**Allocation**

- FCFS
- RR
- DPF

**Latency**

- FCFS
- DPF N=75
- DPF N=375

Number of pipelines allocated vs. N parameter for DPF and RR

Fraction of pipelines (CDF) vs. Pipeline scheduling delay with Timeout
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


References


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