OpenDP

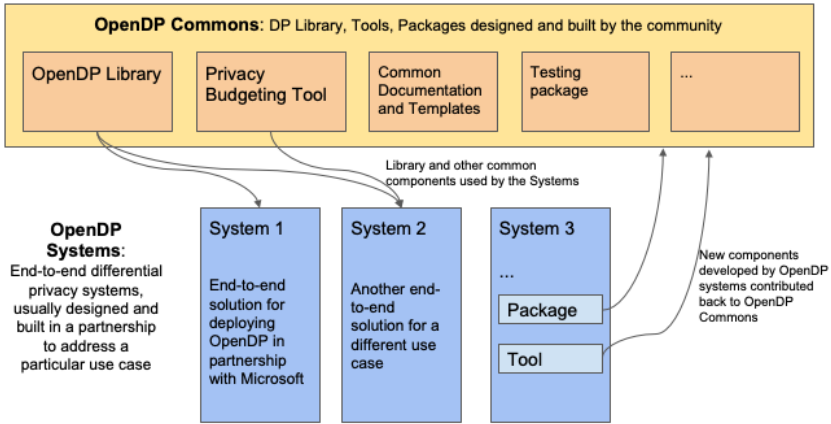
Systems Integrations

Joshua Allen

James Honaker

May 13, 2020

OpenDP: An Open-Source platform for Differential Privacy



Library desiderata $\Rightarrow$ System Principles

- **Extensibility**
- **Flexibility**
- **Verifiability**
- **Programmability**
- **Modularity**
- **Usability**
- **Efficiency**
- **Utility**

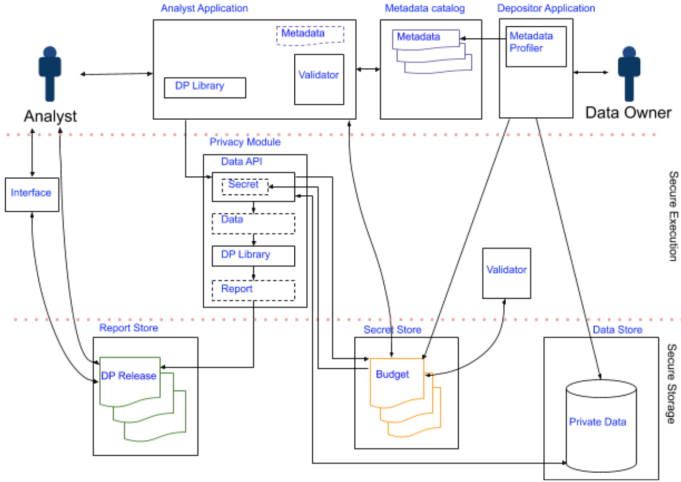
Key Questions

- What types of stores need Commons components integrate with?
- What types of **identity**, **authentification** and **authorization** systems will Commons integrate with?
- What are the use cases utilitizing distributed data/distributed computation?
- Are there limitations that language choices for library impose on systems?
- What is the best API for library adoption? (language neutral? unopinionated?)

Example System Joint with Microsoft

- This system is an end-to-end platform to allow DP queries and statistical releases to senstive data.
- We provide several basic building blocks that can be used by people involved with sensitive data, with implementations based on vetted and mature differential privacy research.
- We provide a pluggable open source framework that researchers can use to try newer mechanisms and algorithms.

Example System Joint with Microsoft



Components

Current Component List:

1. **Library** of DP algorithms, components, utilities
2. Analysis Graph **API**
3. **SDK** support to execute against CSV, SQL Server, PostgreSQL, Dataverse, in Azure
4. **SQL** an ODBC-style API to produce differentially private reports from SQL queries (SQL-92) , focused on common aggregates used in analytics.
5. **Static Validator** certify a proposed analysis is DP and compose privacy loss
6. **Stocastic Validator** test workflows numerically against synthetic data for privacy violations

Library Language Choice

Library implemented in Rust:

- Rust is a fast-growing, imperative functional language.
- high performance, reliable, good memory safety and type safety.
- structures, generics, higher-order functions, and values are immutable by default.
- It has a strong ownership model that gives memory safety.

Collectively, this makes Rust code easier to reason about concretely.

Library Language Choice

Library implemented in Rust:

- Rust is a fast-growing, imperative functional language.
- high performance, reliable, good memory safety and type safety.
- structures, generics, higher-order functions, and values are immutable by default.
- It has a strong ownership model that gives memory safety.

Collectively, this makes Rust code easier to reason about concretely.

Proposal: Rust should be the language for the robust, polished, production code that enters the maximally trusted OpenDP library. However, to encourage contributions from researchers and other developers, the library should be able to work with Python and R code components.

Exploratory Analysis

- API to the library defines an analysis as a computational graph.

Exploratory Analysis

- API to the library defines an analysis as a computational graph.
- Proposed analysis is validated for privacy preservation.
- If certified, sent to the runtime.

Exploratory Analysis

- API to the library defines an analysis as a computational graph.
- Proposed analysis is validated for privacy preservation.
- If certified, sent to the runtime.
- Python and R bindings allow easy access to specify an analysis through the API.

```
with system.Analysis() as analysis:
    data = op.Dataset(path = data_path, column_names = var_names)

    income_histogram = op.dp_histogram(
            op.cast(data['income'], type='int', lower=0, upper=100),
            edges = income_edges,
            upper = 1000,
            null_value = 150,
            privacy_usage = {'epsilon': 0.5}
    )

analysis.release()
```

In [2]:

```
income_edges = list(range(0, 100000, 10000))
education_categories = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16"]


with system.Analysis() as analysis:
    data = op.Dataset(path = data_path, column_names = var_names)

    income_histogram = op.dp_histogram(
            op.cast(data['income'], type='int', lower=0, upper=100),
            edges = income_edges,
            upper = 1000,
            null_value = 150,
            privacy_usage = {'epsilon': 0.5}
        )

    income_prep = op.histogram(op.cast(data['income'], type='int', lower=0, upper=100),
                               edges=income_edges, null_value =-1)
    income_histogram2 = op.laplace_mechanism(income_prep, privacy_usage={"epsilon": 0.5, "delta": .000001})

    sex_histogram = op.dp_histogram(
            op.cast(data['sex'], type='bool', true_label="0"),
            upper = 1000,
            privacy_usage = {'epsilon': 0.5}
        )

    sex_prep = op.histogram(op.cast(data['sex'], type='bool', true_label="0"), null_value = True)
    sex_histogram2 = op.laplace_mechanism(sex_prep, privacy_usage={"epsilon": 0.5, "delta": .000001})

    education_histogram = op.dp_histogram(
            data['educ'],
            categories = education_categories,
            null_value = "-1",
            privacy_usage = {'epsilon': 0.5}
        )

    education_prep = op.histogram(data['educ'],
            categories = education_categories, null_value = "-1")
    education_histogram2 = op.laplace_mechanism(education_prep, privacy_usage={"epsilon": 0.5, "delta": .000001})


analysis.release()

print("Income histogram Geometric DP release:   " + str(income_histogram.value))
print("Income histogram Laplace DP release:     " + str(income_histogram2.value))

print("Sex histogram Geometric DP release:      " + str(sex_histogram.value))
print("Sex histogram Laplace DP release:        " + str(sex_histogram2.value))

print("Education histogram Geometric DP release:" + str(education_histogram.value))
print("Education histogram Laplace DP release:  " + str(education_histogram2.value))
```

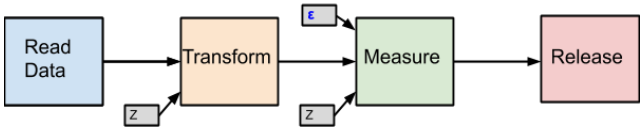
```
Income histogram Geometric DP release:   [337 198 137  78  46  60  61  69  58  77]
Income histogram Laplace DP release:     [328.17803145 189.80390631 122.94969197  96.43640724  60.06534022
  49.54764655  46.47162469  19.61143614  16.69023222  67.87543335]
```

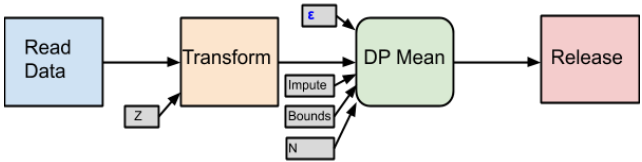
API

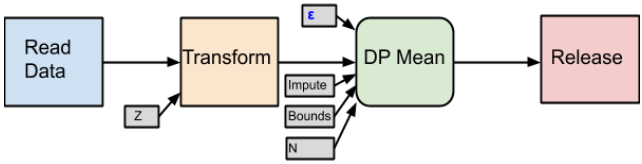
- We built a library API using Protocol Buffers,
  - serialized data language, like JSON or XML, but heavily typed, very compact.
- Rust uses same Foreign Function Interfaces as C++ so bindings direct to Rust are identical (cleaner to write),
- API provides bindings to Python and R,
- For Python, up on PyPI and pip installable.

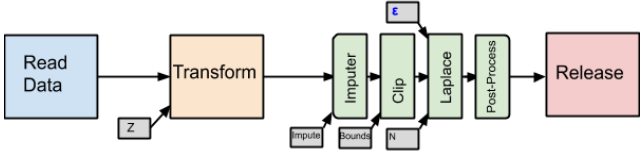
API Exemplars

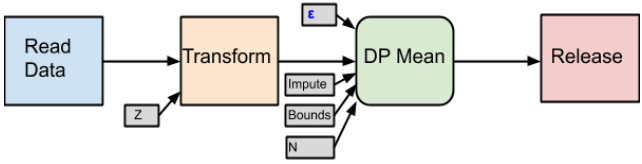
The following contain examples of the computational graph representation of an analysis (or plan) for a proposed release.

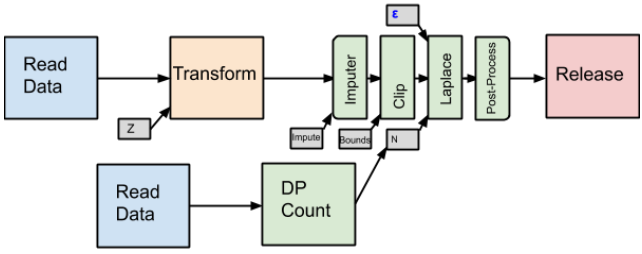

Read Data
Transform
Measure
Release
ε
Z
Z

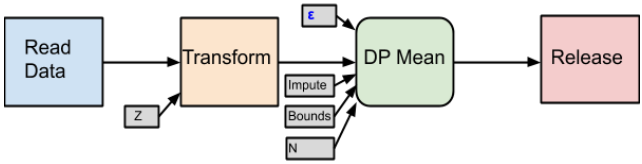

Read Data
Transform
Z
ε
Impute
Bounds
N
DP Mean
Release

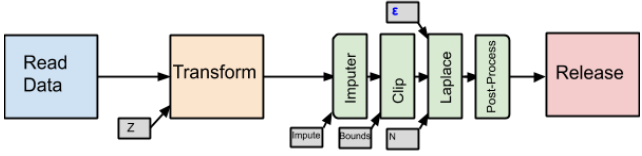

Read Data
Transform
Z
ε
Impute
Bounds
N
DP Mean
Release

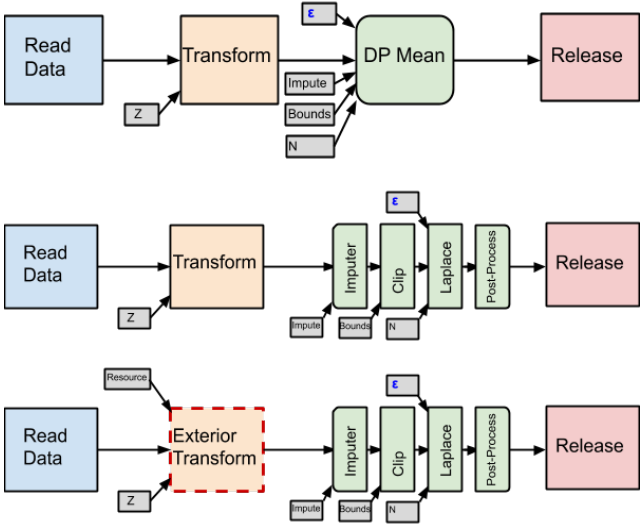

Read Data
Transform
Z
ε
Impute
Bounds
N
Imputer
Clip
Laplace
Post-Process
Release

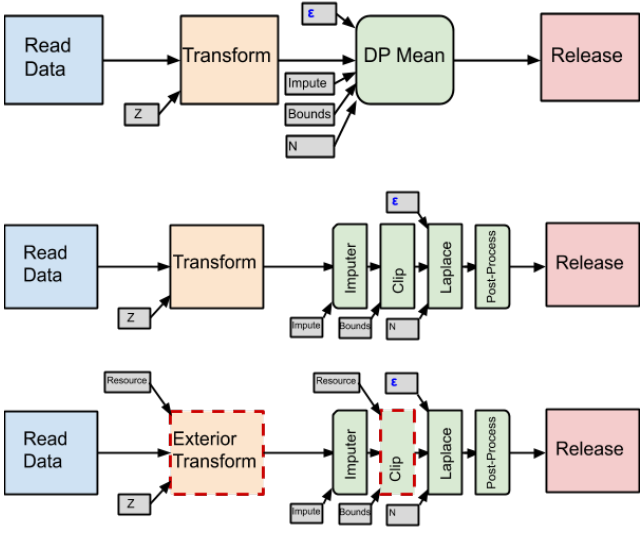

Read Data
Transform
Z
ε
DP Mean
Impute
Bounds
N
Release


Read Data
Transform
Z
Imputer
Impute
Clip
Bounds
ε
Laplace
N
Post-Process
Release
Read Data
DP Count

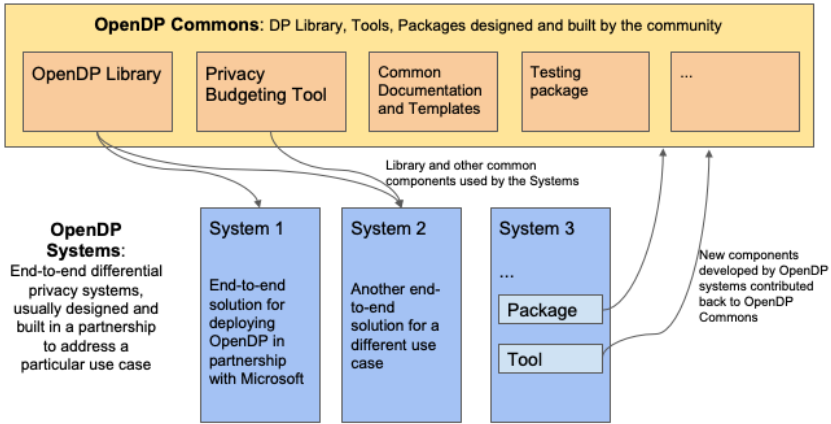

Read Data
Transform
Z
ε
Impute
Bounds
N
DP Mean
Release

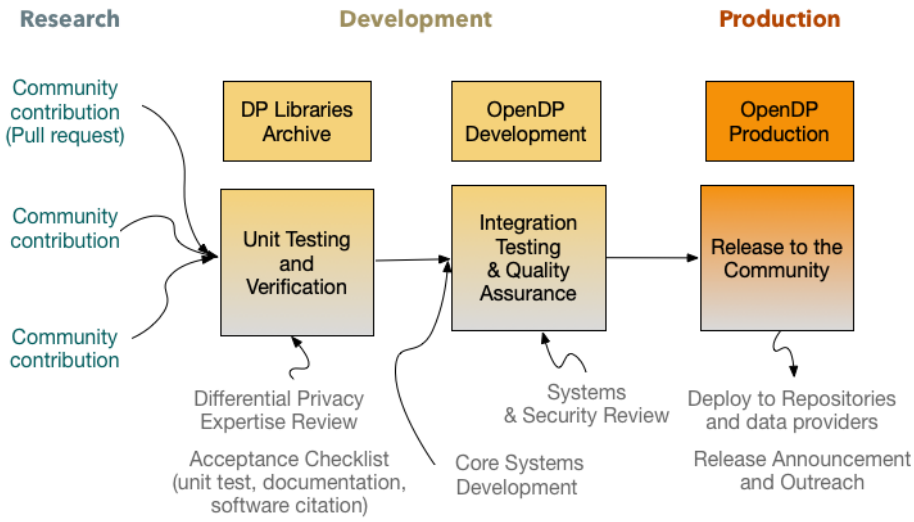

Read Data
Transform
Z
ε
Impute
Bounds
N
Imputer
Clip
Laplace
Post-Process
Release


Read Data
Transform
DP Mean
Release
ε
Z
Impute
Bounds
N
Read Data
Transform
Imputer
Clip
Laplace
Post-Process
Release
ε
Z
Impute
Bounds
N
Resource
Read Data
Exterior Transform
Imputer
Clip
Laplace
Post-Process
Release
ε
Z
Impute
Bounds
N


Read Data
Transform
DP Mean
Release
ε
Z
Impute
Bounds
N
Read Data
Transform
Imputer
Clip
Laplace
Post-Process
Release
ε
Z
Impute
Bounds
N
Resource
Read Data
Exterior Transform
Imputer
Clip
Laplace
Post-Process
Release
Resource
ε
Z
Impute
Bounds
N

OpenDP: An Open-Source platform for Differential Privacy




Research
Development
Production
Community contribution (Pull request)
Community contribution
Community contribution
DP Libraries Archive
OpenDP Development
OpenDP Production
Unit Testing and Verification
Integration Testing & Quality Assurance
Release to the Community
Differential Privacy Expertise Review
Acceptance Checklist (unit test, documentation, software citation)
Core Systems Development
Systems & Security Review
Deploy to Repositories and data providers
Release Announcement and Outreach

Key Questions

- What types of stores need Commons components integrate with?
- What types of **identity**, **authentification** and **authorization** systems will Commons integrate with?
- What are the use cases utilitizing distributed data/distributed computation?
- Are there limitations that language choices for library impose on systems?
- What is the best API for library adoption? (language neutral? unopinionated?)