



# Flexible Network Analytics in the Cloud

**CHI-NOG 07**

May 18, 2017

**Jon M. Dugan**

Software Engineering Group  
ESnet (Energy Sciences Network)  
Lawrence Berkeley National Lab



Chicago Network  
Operators Group



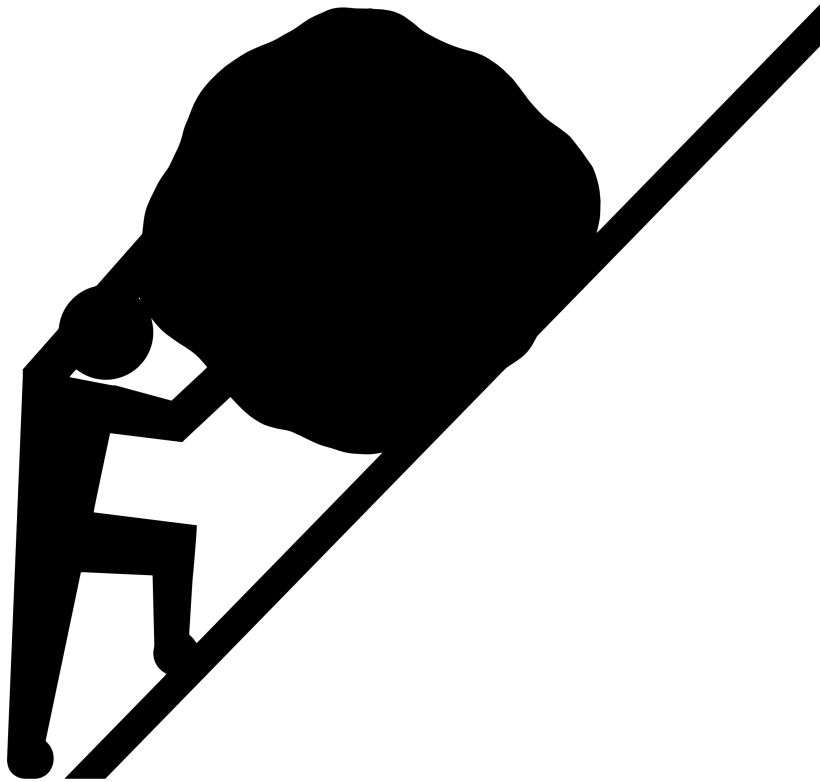
**ESnet**  
ENERGY SCIENCES NETWORK



U.S. DEPARTMENT OF  
**ENERGY**

Office of  
Science

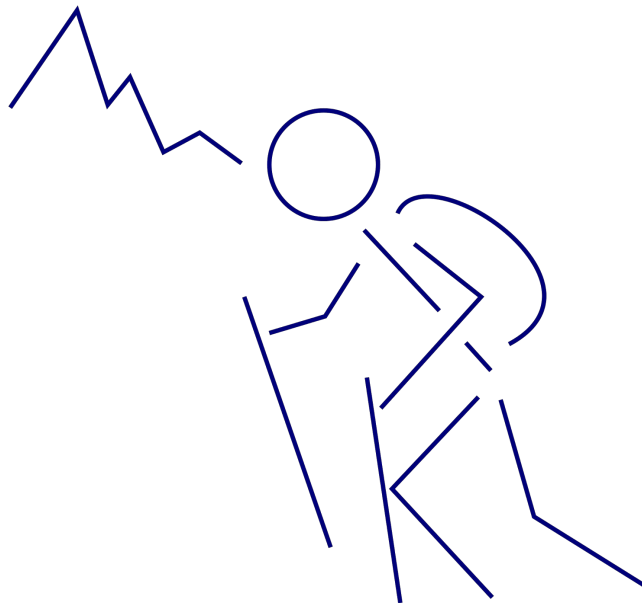
# Sisyphus, Network Engineer



# Crazy Happy Kid, Network Engineer



# Rugged Adventurer, Network Engineer



# Outline

Hard realities

Coping strategies

A programming model brings solace

Making the ascent with code

A scenic vista

# Hard realities

Network measurement is harder than it seems.

# Reality #1: It's a mess

Everything is a struggle.

# Ideal





# Reality



The world is not neat and tidy.

Your data is not neat and tidy.

Accept it, design for it, live with it.  
Invite it over for dinner.

# Reality #2: Things Change

I'm not kidding, everything *is* a struggle.



# First Request





# Second Request



What you needed yesterday probably  
isn't what you'll need tomorrow.

You have to build something today  
that will work tomorrow.

Don't make decisions today that you can't unmake  
tomorrow.

Save your raw data. All of it if possible.

# Reality #3: There's always more

I told you everything is a struggle, but I'm not sure you believe me.



Photo: Moscow. Standard buildings on Novokosinskaya street, аналоговый снимок via Wikimedia Commons.



There are always new devices coming online  
and more telemetry you can collect.

Your storage and compute will be the same size  
tomorrow as they are today.

Use the cloud to scale.  
The cloud is elastic and (effectively) infinite.

# Reality #4: It's never really done

Nevermind. You'll see for yourself, everything is a struggle.

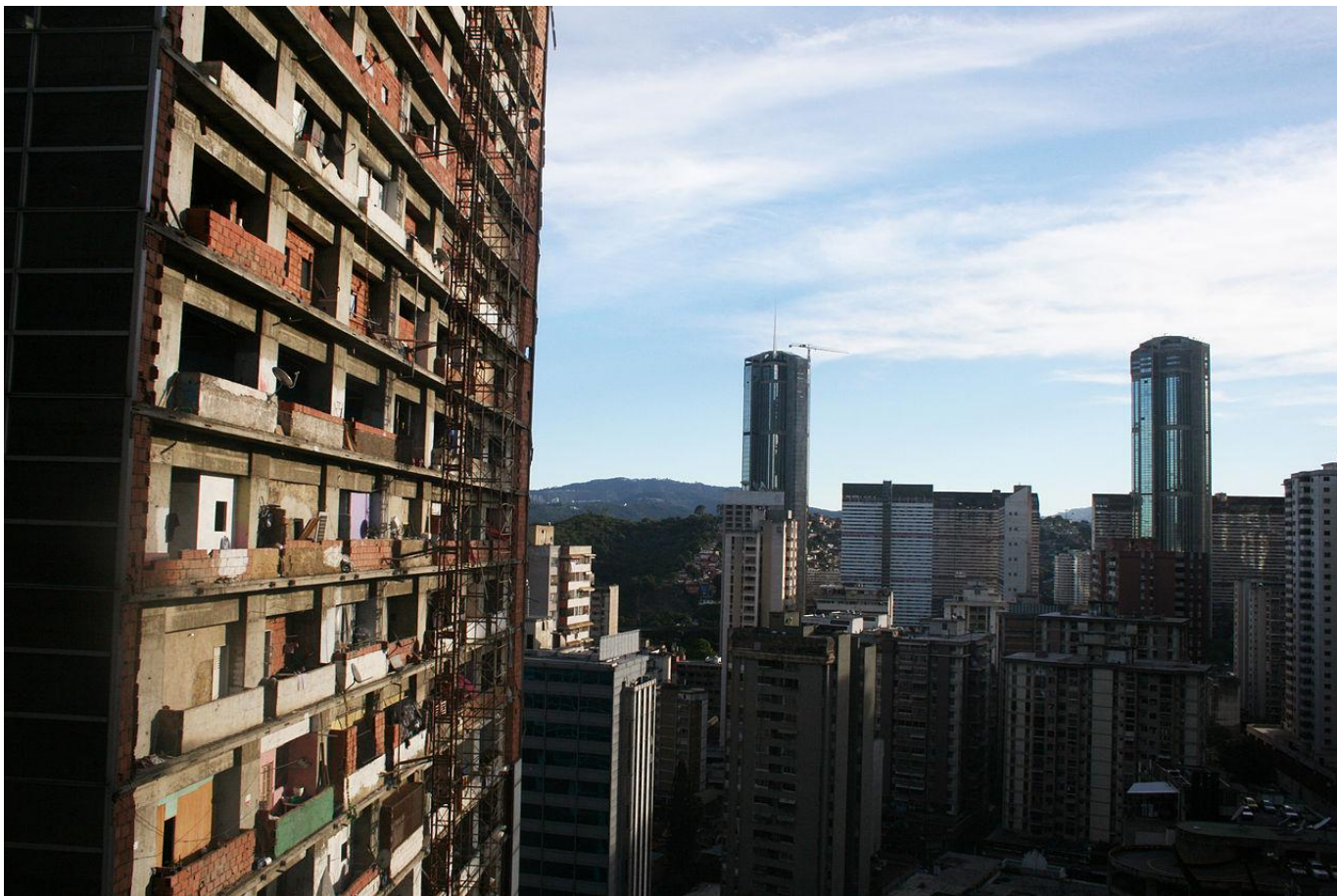


Photo: © Saúl Briceño from Wikimedia Commons Exterior of Centro Financiero Confinanzas (Torre de David),

Nothing is ever really finished.

Time and money are limited.

Spend your time on the “what”: deriving insight  
from your data.  
...not the “how”: building infrastructure.

# Coping strategies

# The reality of the situation

## 1. It's a mess

- Design knowing things won't be tidy

## 3. There's always more

- Rely on the cloud for scaling

## 2. Things change

- Keep raw data to keep your options open

## 4. It's never really done

- “What” not “How”

A programming model brings  
solace

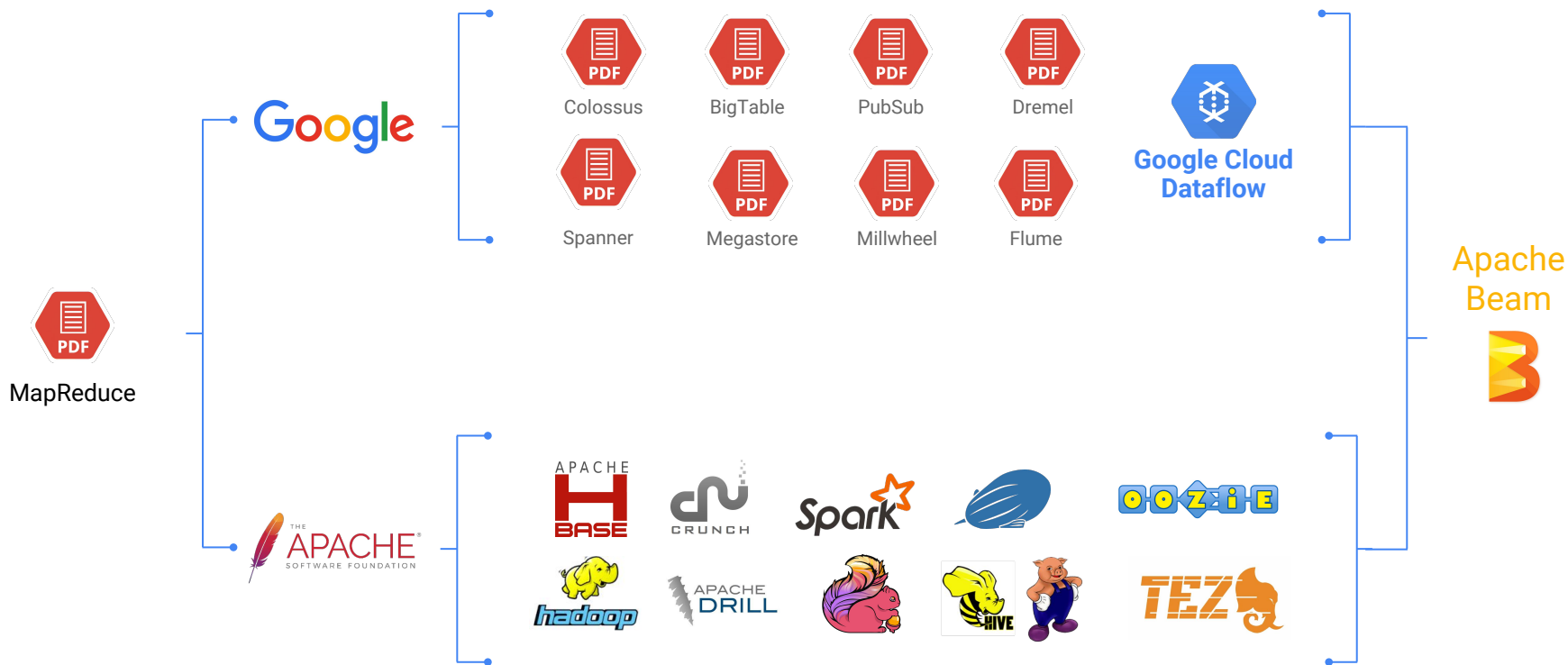
# What is Apache Beam?

1. The Beam Programming Model
2. SDKs for writing Beam pipelines
3. Runners for existing distributed processing backends
  - Apache Apex
  - Apache Flink
  - Apache Spark
  - Google Cloud Dataflow
  - Local runner for testing





# The Evolution of Apache Beam



# Why Apache Beam?

***Unified*** - One model handles batch and streaming use cases.

***Portable*** - Pipelines can be executed on multiple execution environments, avoiding lock-in.

***Extensible*** - Supports user and community driven SDKs, Runners, transformation libraries, and IO connectors.



# The Beam Model: Asking the Right Questions

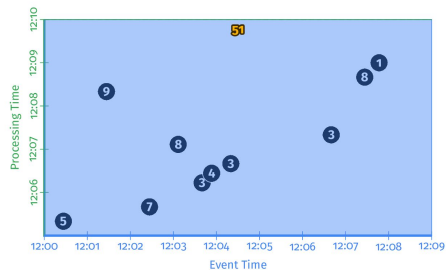
**What** results are calculated?

**Where** in event time are results calculated?

**When** in processing time are results materialized?

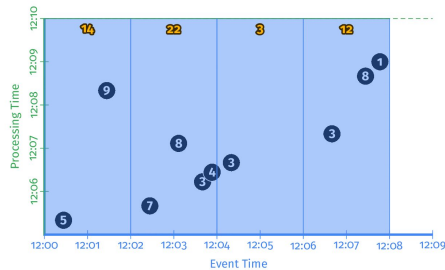
**How** do refinements of results relate?

# Customizing **What** **Where** **When** **How**



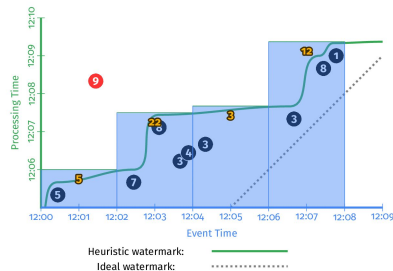
1

**Classic  
Batch**



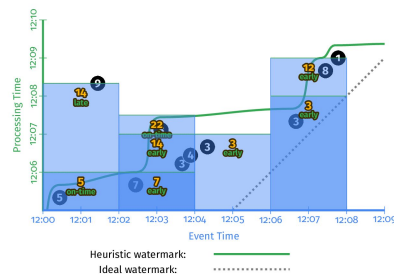
2

**Windowed  
Batch**



3

**Streaming**



4

**Streaming  
+ Accumulation**

# Making the ascent with code

# Immutable data store of raw data

Make an immutable (read-only) repository of raw data

- Minimize processing of the data
- Get it into the database as soon as possible
- Keep your raw data for as long as you can

Benefits

- You get to change your mind later
- If you make mistakes in later steps: recompute.

# Views: how to turn raw data into usable data

Use Beam to transform your raw data into a “view” of the data

Examples:

- converting SNMP counters to rates
- grouping SNMP counters by customer or peer
- binning flows by attributes
- building network topologies from basic data

Views can be stacked, you can use one view as the input for a more refined view.

# Core Beam Abstractions

## PCollections

- Distributed, multi-element data set.

## Transforms

- Some code to run on all the data
- Take PCollection in and produce a PCollection

## Pipeline I/O

- Read data and produce a PCollection
- Take a PCollection and write data

## Pipeline

- $\text{Input} \rightarrow \text{PCollection} \rightarrow \text{Transforms} \rightarrow \text{PCollection} \rightarrow \text{Output}$



# $f(\text{allMyData})$ : Focus on “what”, not “how”

Define the “what”

- I/O: What data to use and where it lives.
- PCollections: Grouping the data for each transformation step
- Transforms: allow you to define the “what”

Let Beam handle the “how”

- I/O: Reading and writing data at scale
- PCollections: distributing data across workers
- Transforms: applying your analysis to PCollections

# Example: Computing Total Traffic

```
# Python Beam SDK
pipeline = beam.Pipeline('DirectRunner')

(pipeline
 | 'read'          >> ReadFromText('./example.csv')
 | 'csv'           >> beam.ParDo(FormatCSVDoFn())
 | 'ifName key'    >> beam.Map(group_by_device_interface)
 | 'group by iface' >> beam.GroupByKey()
 | 'compute rate'  >> beam.FlatMap(compute_rate)
 | 'timestamp key' >> beam.Map(lambda row: (row['timestamp'], row['rateIn']))
 | 'group by timestamp' >> beam.GroupByKey()
 | 'sum by timestamp' >> beam.Map(lambda rates: (rates[0], sum(rates[1])))
 | 'format' >> beam.Map(lambda row: '{}{}'.format(row[0], row[1]))
 | 'save' >> beam.io.WriteToText('./total_by_timestamp'))

pipeline.run()
```

# Data Processing Tradeoffs

**1+1=2**

**Completeness**



**Latency**



**Cost**

# Stream or Batch?

## Streaming for *realtime insight*

- Current network load
- Operational awareness
- Threat detection / Anomaly Detection
- Tend to be more expensive (VMs always running)
- Unbounded data sets

## Batch for *precision or results that can wait*

- Billing
- Precise traffic reports
- Capacity planning
- Tend to be cheaper (VMs used in bursts)
- Bounded data sets

A scenic vista

# What can we see from here?

- Define new views to get different perspectives on raw data
- Add new immutable data sets to gain more dimensions of data (SNMP, Flow/sFlow, syslog, ...)
- Try out new ideas (running batch jobs is cheap and doesn't impact system)
- Teach others to write their own analysis

# Our experiences so far

1. There is a learning curve.
2. Docs aren't amazing, but getting there.
3. You may have to adjust your thinking. Need to understand the model to know what will work at scale.
4. The cloud providers have a several choices when it comes to databases. It's easy to spend a lot of time investigating.
5. Cost is manageable but it's good to keep an eye on it.
6. In the interest of vendor neutrality details about our specific vendor haven't been covered, but I'm happy to talk to you after the talk.

# Thank you!

## Questions?



Jon Dugan <[jdugan@es.net](mailto:jdugan@es.net)>

<http://x1024.net/blog/2017/05/chinog-flexible-network-analytics-in-the-cloud/>

**More about Apache Beam:**

<https://beam.apache.org>

**The World Beyond Batch 101 & 102**

<https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101>

<https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-102>

**More about ESnet:**

Open source: <http://software.es.net/>

Visualization portal: <https://my.es.net/>

