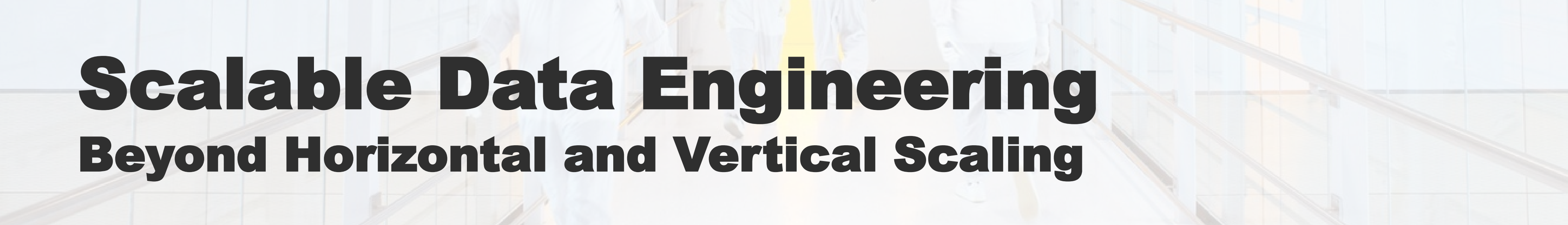
GlobalFoundries®



Scalable Data Engineering

Beyond Horizontal and Vertical Scaling

May-2026

Dr. Paul Jungmann


Dr. Paul Jungmann

Agenda

Intro to GlobalFoundries and Semiconductor Fabrication

Beyond Horizontal & Vertical Scaling

Example: Scalable Virtual Metrology

Summary


Dr. Paul Jungmann

Agenda

Intro to GlobalFoundries and Semiconductor Fabrication

Beyond Horizontal & Vertical Scaling

Example: Scalable Virtual Metrology

Summary

Introduction to GlobalFoundries and Semiconductor Fabrication

Silicon Saxony – High tech as a job engine! 2,500 companies with 70,500 employees

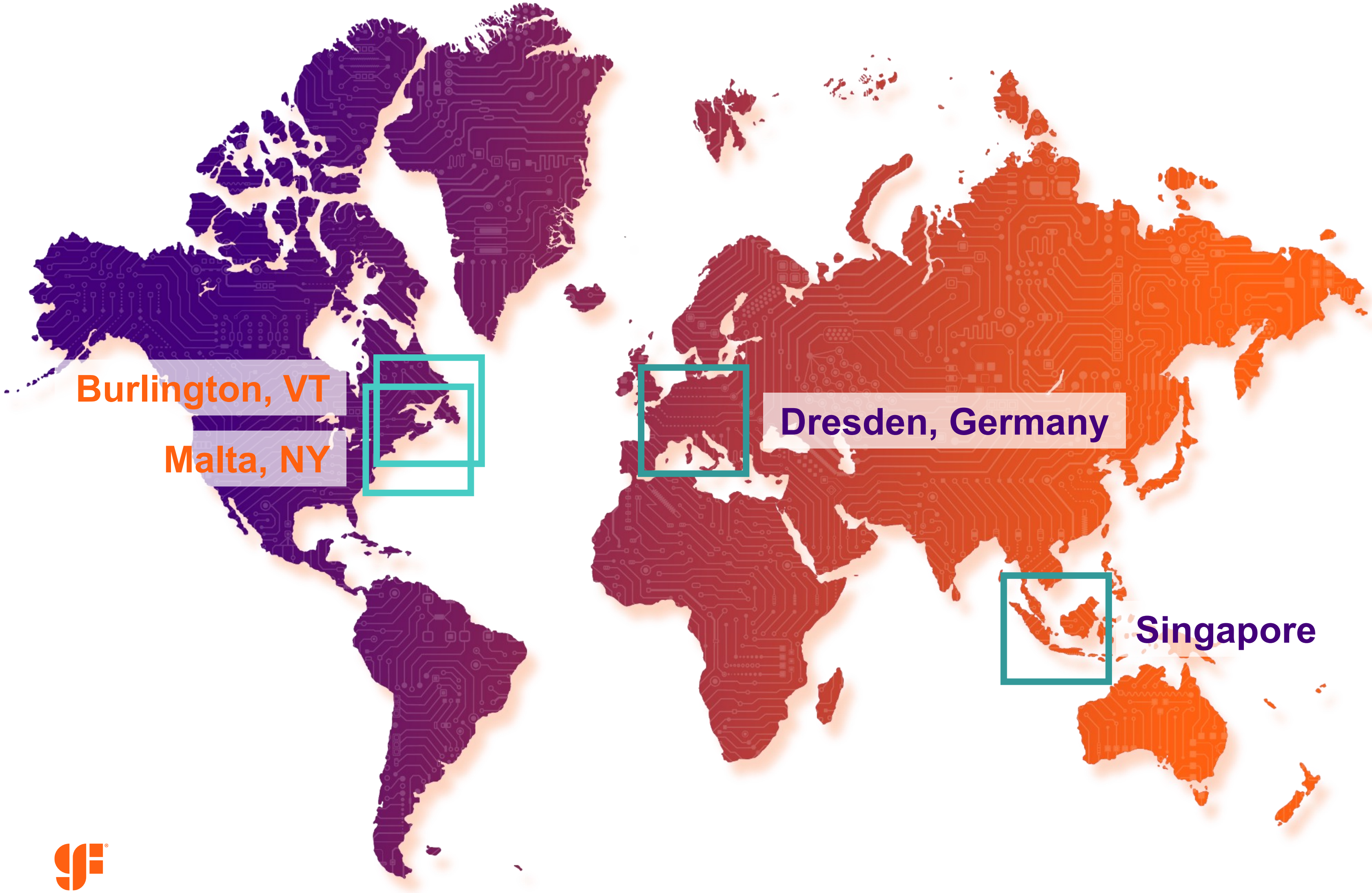
Introduction to GlobalFoundries and Semiconductor Fabrication

GF Dresden is the driving force behind Silicon Saxony, Europe's No. 1 chip cluster.

© Silicon Saxony

Introduction to GlobalFoundries and Semiconductor Fabrication

"Global" is in our nature, since 2009



Technologies

Burlington:
90nm, 180nm

Dresden:
22nm, 28nm, 40nm, 55nm

Malta:
12nm, 22nm

Singapore:
28nm, 40nm, 55nm

Introduction to GlobalFoundries and Semiconductor Fabrication

GF Dresden: Who we are and what we do

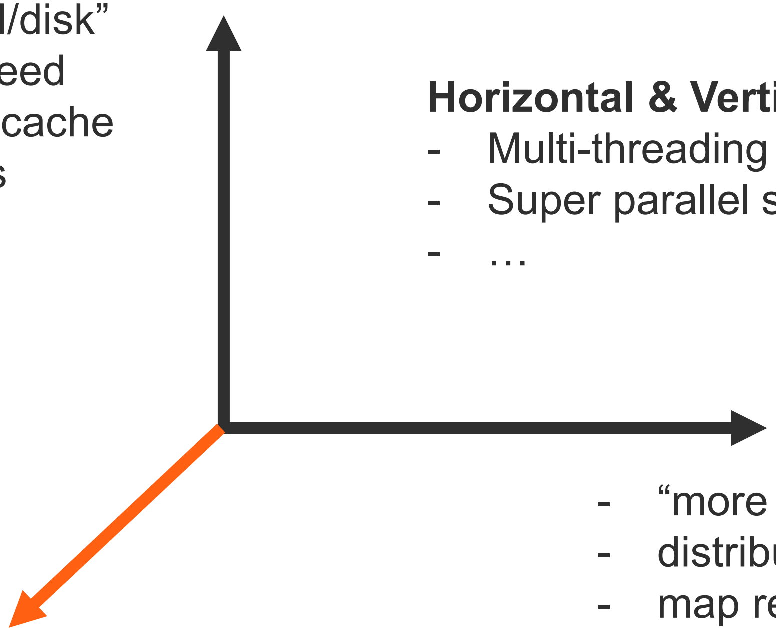
- Contract manufacturer for the global chip industry since 1996/2009
- Approx. 3,000 employees from 50 countries
- Approx. 60,000 m² of clean rooms and laboratories
- Production and R&D for a strong technology portfolio
- Chips for cars, industrial goods, smart consumer devices, etc.
- Trusted chips for A&D, passports, encryption, banking
- Driving force behind the Silicon Saxony cluster

Dr. Paul Jungmann

Agenda

- Intro to GlobalFoundries and Semiconductor Fabrication
- **Beyond Horizontal & Vertical Scaling**
- Example: Scalable Virtual Metrology
- Summary

Beyond Horizontal & Vertical Scaling



Vertical Scaling
- “more CPU/RAM/disk”
- Higher Clock Speed
- Larger L1/L2/L3 cache
- Deeper pipelines
- …

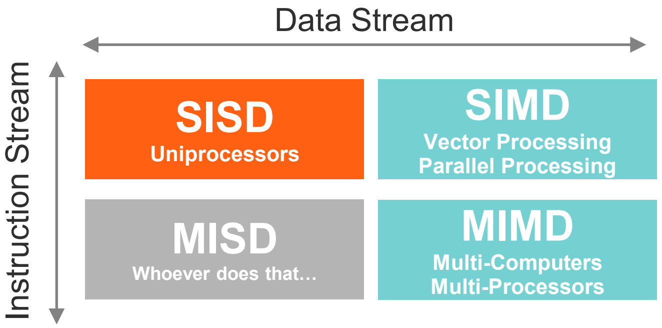
Horizontal & Vertical Scaling
- Multi-threading
- Super parallel super pipelining
- …

Horizontal Scaling
- “more Compute nodes”
- distributed computing
- map reduce
- …

Application Scaling

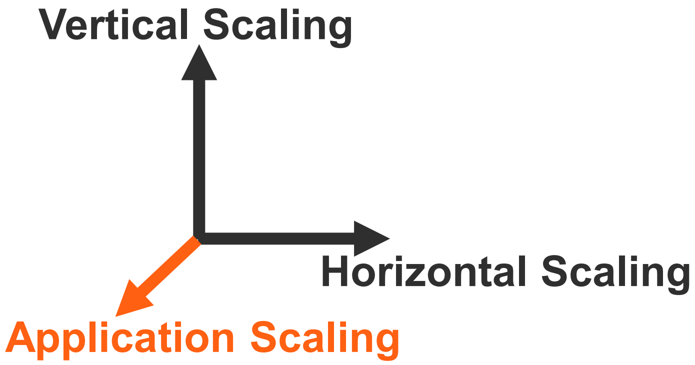
→ **Easy setup of similar (but not the same) applications**

Recall: Flynn Classification



Instruction Stream ↕ / Data Stream ↔		
	SISD Uniprocessors	**SIMD** Vector Processing Parallel Processing
	MISD Whoever does that…	**MIMD** Multi-Computers Multi-Processors

Beyond Horizontal & Vertical Scaling


Vertical Scaling

Horizontal Scaling

Application Scaling

What if there are hundreds of similar applications are to be deployed?

- In the same system/environment?
- Across different systems/environments?
- With inconsistent data sources?

Data Engineering in Application Scaling

Specific for 1 Application

Pro	Con
Fast initial development	Redevelopment for each application
Lighter/Faster code	Hard to maintain

→ Usually no option

How to find the sweet spot?

Generic for all Applications

Pro	Con
Made to be reused	Hard to develop
Better Maintenance	Bloated code, special cases

→ Usually no option

Beyond Horizontal & Vertical Scaling

Data Engineering in Application Scaling

Specific for 1 Application

Pro	Con
Fast initial development	Redevelopment for each application
Lighter/Faster code	Hard to maintain

How to find the sweet spot?

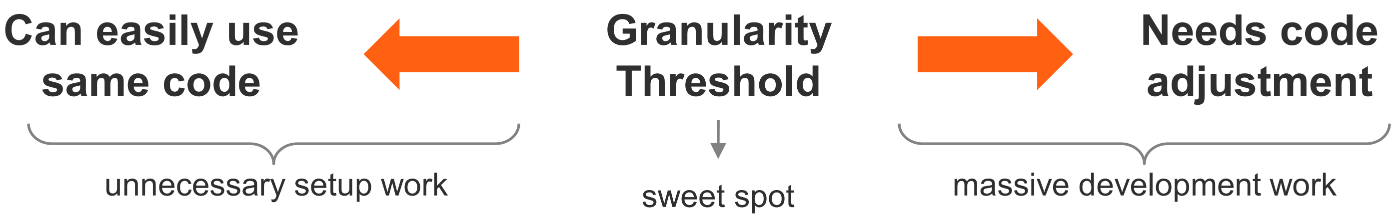
Generic for all Applications

Pro	Con
Made to be reused	Hard to develop
Better Maintenance	Bloated code, special cases

Understanding the sweet spot can be very difficult – it's strongly use case depended

- Understand the use case – talk to domain experts
- Deployment Strategy / Operations Strategy / Maintenance Strategy
- Complexity of the code or pipeline
- Understand or define the use case "granularity":

"Increment" with which paste the applications will be deployed



Can easily use same code

Granularity Threshold

Needs code adjustment

unnecessary setup work

sweet spot

massive development work

Dr. Paul Jungmann

Agenda

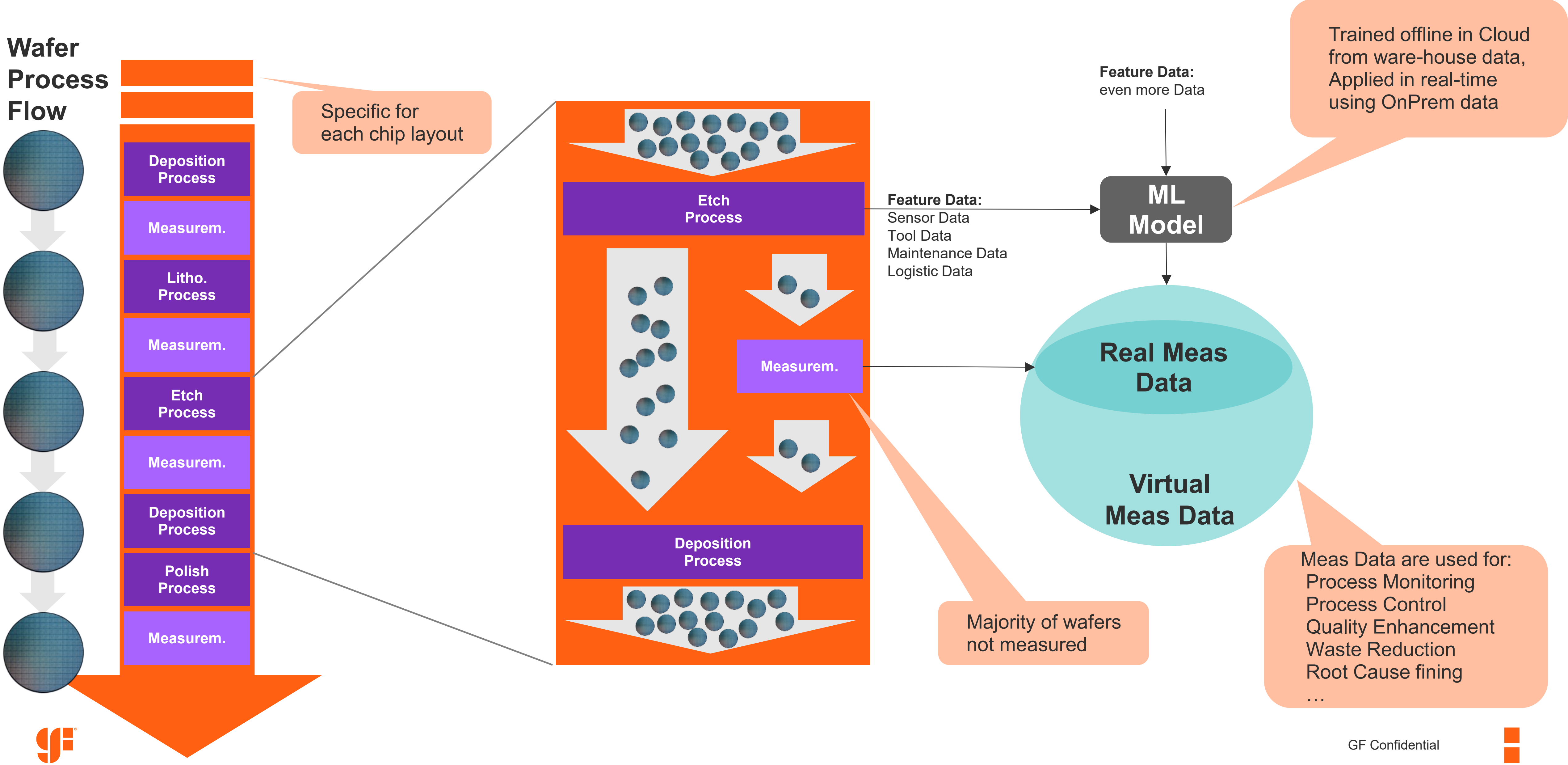
Intro to GlobalFoundries and Semiconductor Fabrication

Beyond Horizontal & Vertical Scaling

Example: Scalable Virtual Metrology

Summary

Example: Scalable Virtual Metrology



Specific for each chip layout

Feature Data:
Sensor Data
Tool Data
Maintenance Data
Logistic Data

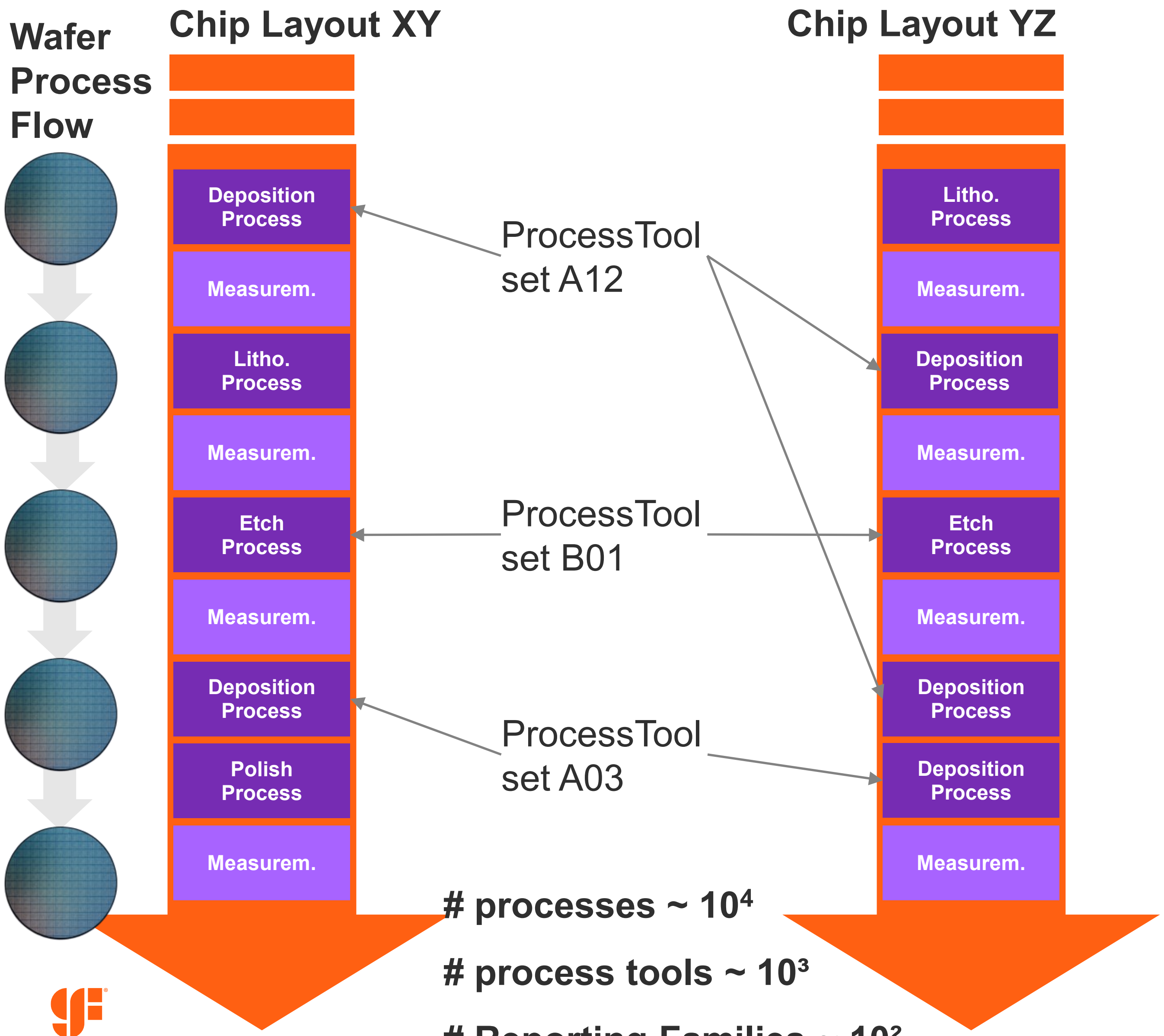
Feature Data:
even more Data

Trained offline in Cloud from ware-house data, Applied in real-time using OnPrem data

Majority of wafers not measured

Meas Data are used for:
- Process Monitoring
- Process Control
- Quality Enhancement
- Waste Reduction
- Root Cause fining
- …

Example: Scalable Virtual Metrology



Wafer Process Flow

Chip Layout XY

Deposition Process
Measurem.
Litho. Process
Measurem.
Etch Process
Measurem.
Deposition Process
Polish Process
Measurem.

Chip Layout YZ

Litho. Process
Measurem.
Deposition Process
Measurem.
Etch Process
Measurem.
Deposition Process
Deposition Process
Measurem.

ProcessTool set A12

ProcessTool set B01

ProcessTool set A03

 # processes ~ 10^4

 # process tools ~ 10^3

 # Reporting Families ~ 10^2

Process Tool set a.k.a „Reporting Family“

- Set of tools that do very similar processes
  → similar physics/chemistry
  → similar dependencies to model
  → Operated by same domain experts
- Set of tools that “report” to the IT system in the very same way
  → granularity threshold for DataEngineering

Can easily use same code

Granularity Threshold

Needs code adjustment

unnecessary setup work

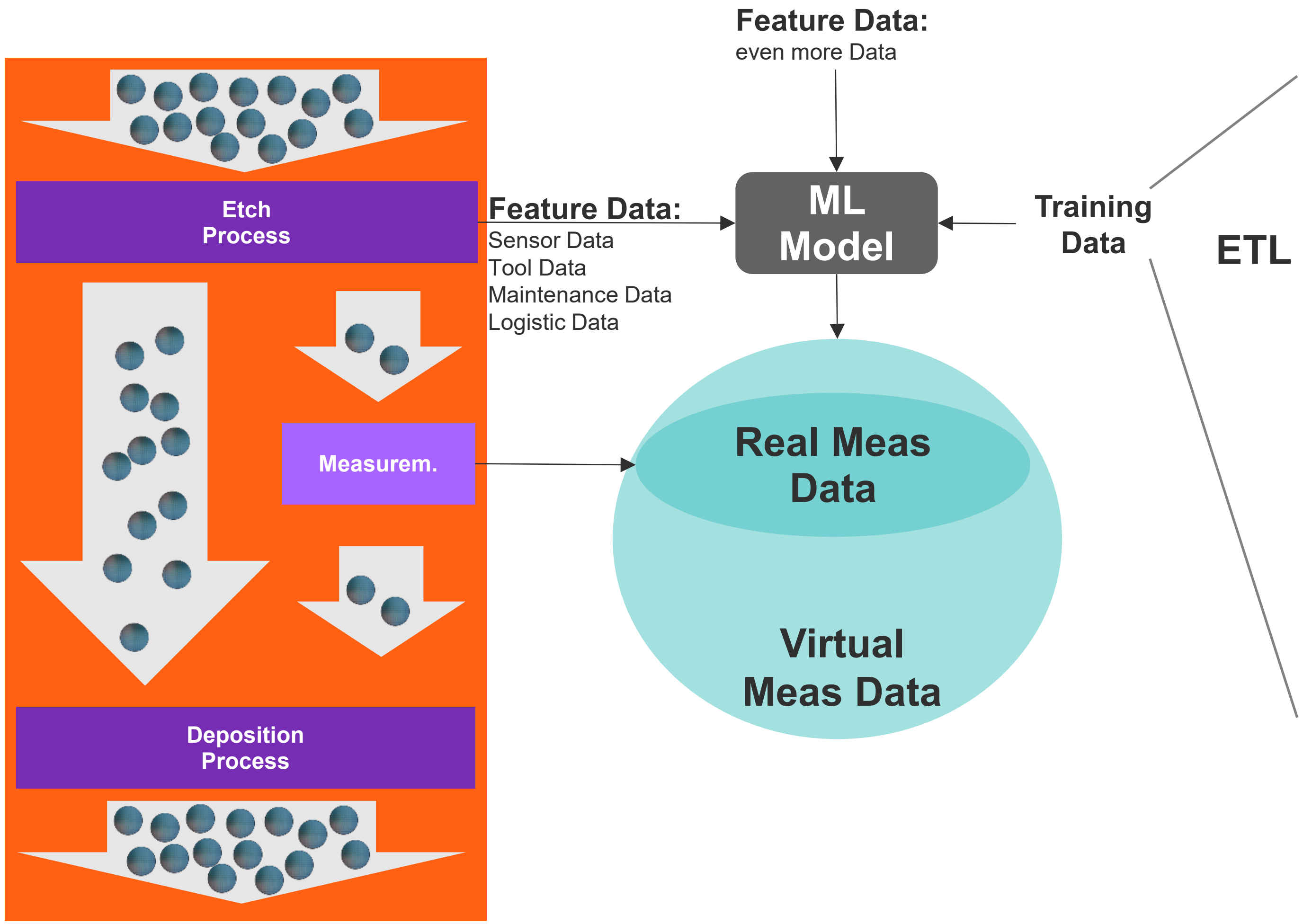
sweet spot

massive development work

- One Application per Reporting Family
  → makes sense from domain perspective
  → fits to the IT system’s structure
  → handable development/setup effort (DataEngineering)
  → overseeable from operational perspective

Example: Scalable Virtual Metrology

1 Lot = defined group of 25 wafers
Site = measurement location on wafer
ETL = Extract, Transfer, Load



Feature Data:
even more Data

Etch Process

Feature Data:
Sensor Data
Tool Data
Maintenance Data
Logistic Data

ML Model

Training Data

ETL

Measurem.

Real Meas Data

Virtual Meas Data

Deposition Process

Data ‚Type‘	Source of training Data	Features /Lables
Aggr. Sensor Data (Wafer lvl)	S3 Bucket	Features
Tool Process Data (Wafer lvl)	RedShift	Features
Tool Process Data (Lot lvl)	RedShift	Features
Real Measurement Data (Site lvl)	RedShift	Labels
Tool Maintenance Data (Tool Event lvl)	S3 Bucket	Features
Tool State Data (Tool Event lvl)	SQL DB	Features
Logistic Data (Wafer/Lot lvl)	multiple	Features

1 Lot = defined group of 25 wafers
Site = measurement location on wafer
ETL = Extract, Transfer, Load

Example: Scalable Virtual Metrology

Data ‚Type‘	Source of training Data	Features /Lables
Aggr. Sensor Data (Wafer lvl)	S3 Bucket	Features
Tool Process Data (Wafer lvl)	RedShift	Features
Tool Process Data (Lot lvl)	RedShift	Features
Real Measurement Data (Site lvl)	RedShift	Labels
Tool Maintenance Data (Tool lvl)	S3 Bucket	Features
Tool State Data (Tool lvl)	SQL DB	Features
Logistic Data (Wafer/Lot lvl)	multiple	Features

DataExtraction straight forward when you know:

- which wafer was processed
  - in which process tool
    - belonging to which Reporting Family
  - at what time

- Which measurement belongs to which process
  - how is the measurement called

What we want/have as input:

- Reporting Family
- Time Frame for training data extraction

Solution:

Make (whenever possible) everything dependent on just ‚Reporting Family‘ and ‚Time Frame‘.
→ get required Meta-Information on the fly during execution

1 Lot = defined group of 25 wafers
Site = measurement location on wafer
ETL = Extract, Transfer, Load

Example: Scalable Virtual Metrology

Data ‚Type‘	Source of training Data	Needed inputs
Aggr. Sensor Data (Wafer lvl)	S3 Bucket	Reporting Family, Time Range
Tool Process Data (Wafer lvl)	RedShift	Wafer List, Process
Tool Process Data (Lot lvl)	RedShift	Wafer List, Process
Real Measurement Data (Site lvl)	RedShift	Wafer List, Measurement to Process
Tool Maintenance Data (Tool lvl)	S3 Bucket	Wafer processing times, Tools
Tool State Data (Tool lvl)	SQL DB	Wafer processing times, Tools
Logistic Data (Wafer lvl)	multiple	Wafer List, Process, Wafer processing times, Measurement to Process

What we want/have as input:

- Reporting Family
- Time Frame for training data extraction

So, make a plan:
which inputs are needed and where to get them

Needed Input	Where to get from
Wafer List	FDC Data and TimeFrame
Tool List	Reporting Family and Logistics DataBase
Reporting Family	given
Time Frame	given
Lookup: Process – Measurement – Parameter	Reporting Family, Logistics DataBases and RedShift

Solution:
Make (whenever possible) everything dependent on just ‚Reporting Family‘ and ‚Time Frame‘.
→ get required Meta-Information on the fly during execution

Beyond Horizontal & Vertical Scaling

Problem:
The inputs the DataSources need don‘t match the inputs the given domain provides.

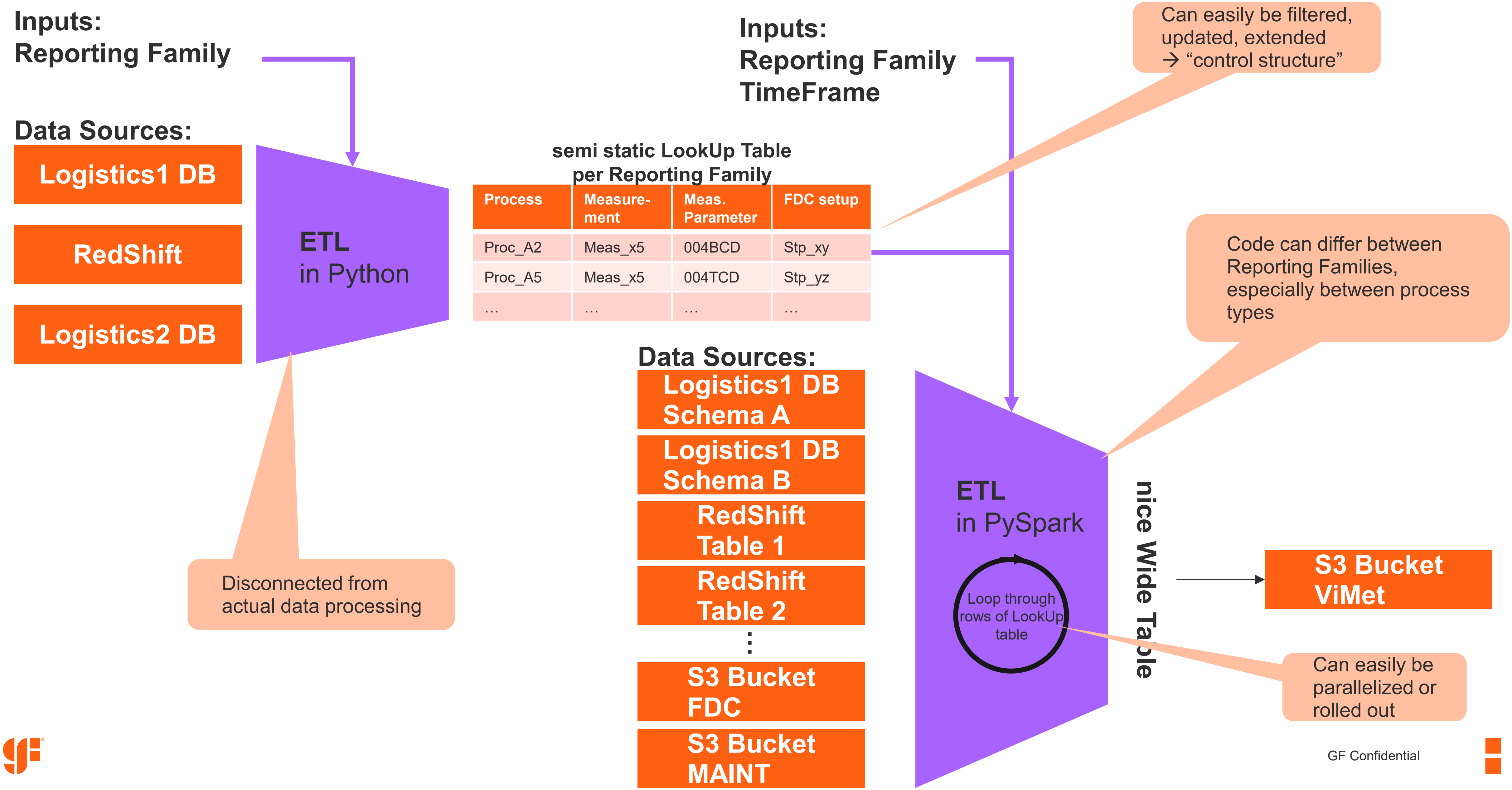
Solution:
Make (whenever possible) everything dependent on just the given inputs.
→ get required Meta-Information on the fly during execution

Don’t overdo it:
Try to understand which data/information are
- fully static → lookup table
- semi static → lookup table with update from time to time
- actually dynamic → assess from data source

Except inconsistencies, especially in historically grown systems
→ Include “control structures” (e.g. lookup table, config file) in the ETL pipeline that can easily be adjusted.
→ Try to decouple the ETL pipeline by these control structures
→ Keep a special view on things where humans interact with the system/data

Package large ETL pipeline into distinct parallel subparts
→ Resilience against system hiccups, easy restart just for “what is left”
→ Can be better to parallelize
→ Problem with one subparts doesn’t affect the whole system

Example: Scalable Virtual Metrology



Inputs:
Reporting Family

Data Sources:

- Logistics1 DB
- RedShift
- Logistics2 DB

ETL
in Python

Disconnected from actual data processing

Inputs:
Reporting Family
TimeFrame

Can easily be filtered, updated, extended
→ "control structure"

semi static LookUp Table per Reporting Family

Process	Measure-ment	Meas. Parameter	FDC setup
Proc_A2	Meas_x5	004BCD	Stp_xy
Proc_A5	Meas_x5	004TCD	Stp_yz
…	…	…	…

Data Sources:

- Logistics1 DB Schema A
- Logistics1 DB Schema B
- RedShift Table 1
- RedShift Table 2
- ⋮
- S3 Bucket FDC
- S3 Bucket MAINT

Code can differ between Reporting Families, especially between process types

ETL
in PySpark

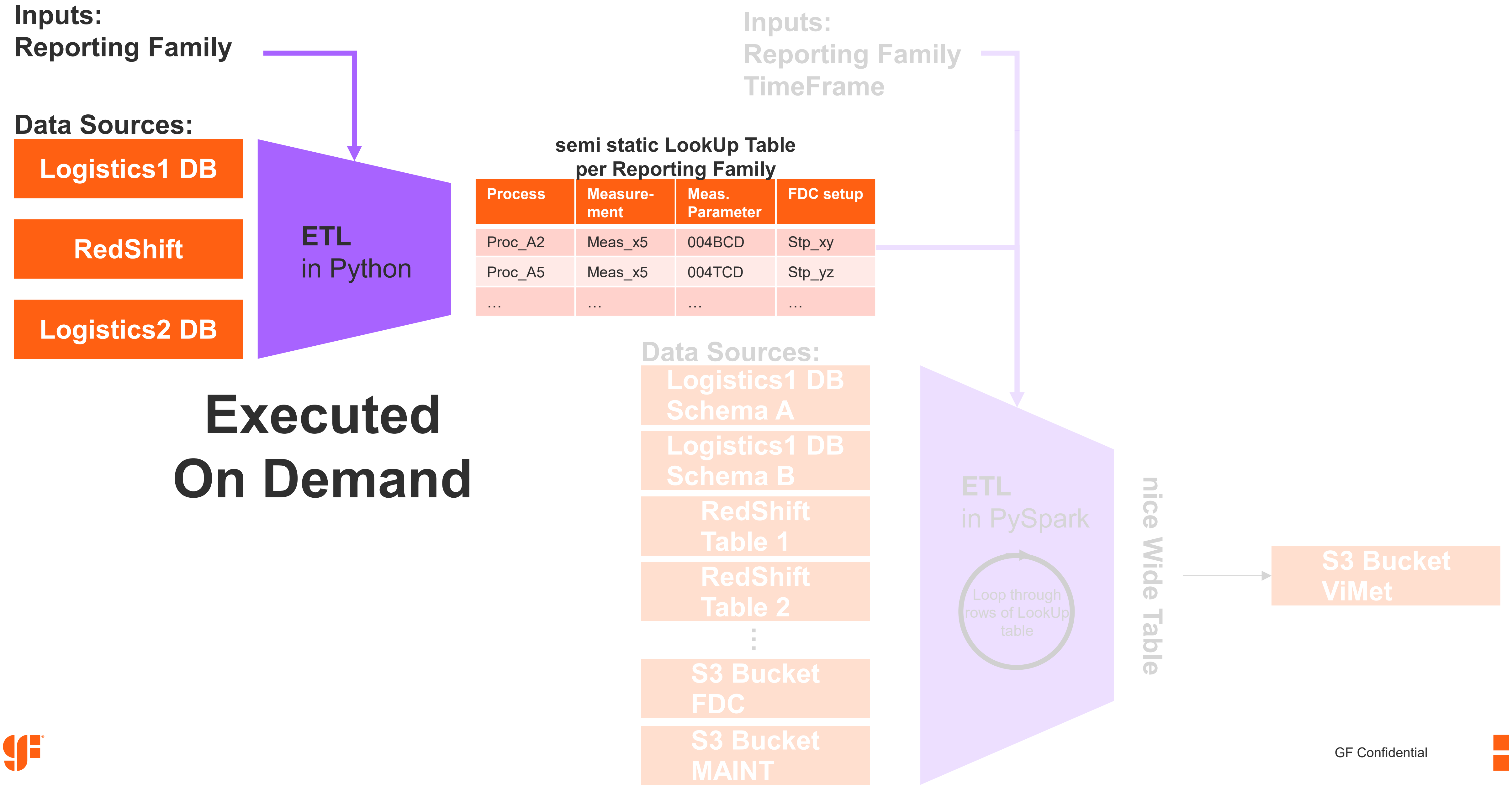
Loop through rows of LookUp table

Can easily be parallelized or rolled out

nice Wide Table

→ S3 Bucket ViMet

Example: Scalable Virtual Metrology



Inputs:
Reporting Family

Data Sources:

Logistics1 DB

RedShift

Logistics2 DB

ETL
in Python

semi static LookUp Table
per Reporting Family

Process	Measure-ment	Meas. Parameter	FDC setup
Proc_A2	Meas_x5	004BCD	Stp_xy
Proc_A5	Meas_x5	004TCD	Stp_yz
…	…	…	…

Executed
On Demand

Inputs:
Reporting Family
TimeFrame

Data Sources:

Logistics1 DB
Schema A

Logistics1 DB
Schema B

RedShift
Table 1

RedShift
Table 2

⋮

S3 Bucket
FDC

S3 Bucket
MAINT

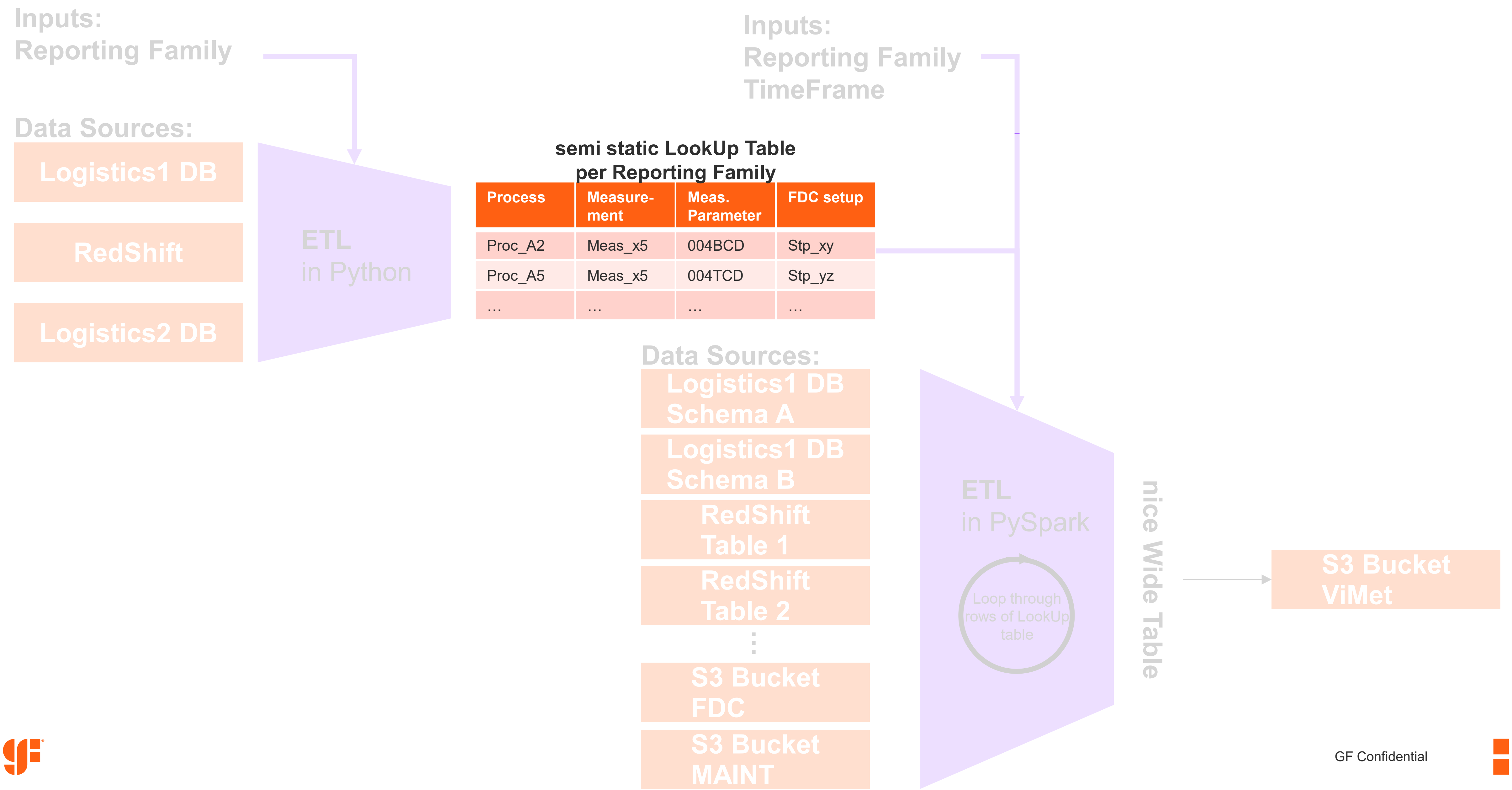
ETL
in PySpark

Loop through rows of LookUp table

nice Wide Table

S3 Bucket
ViMet

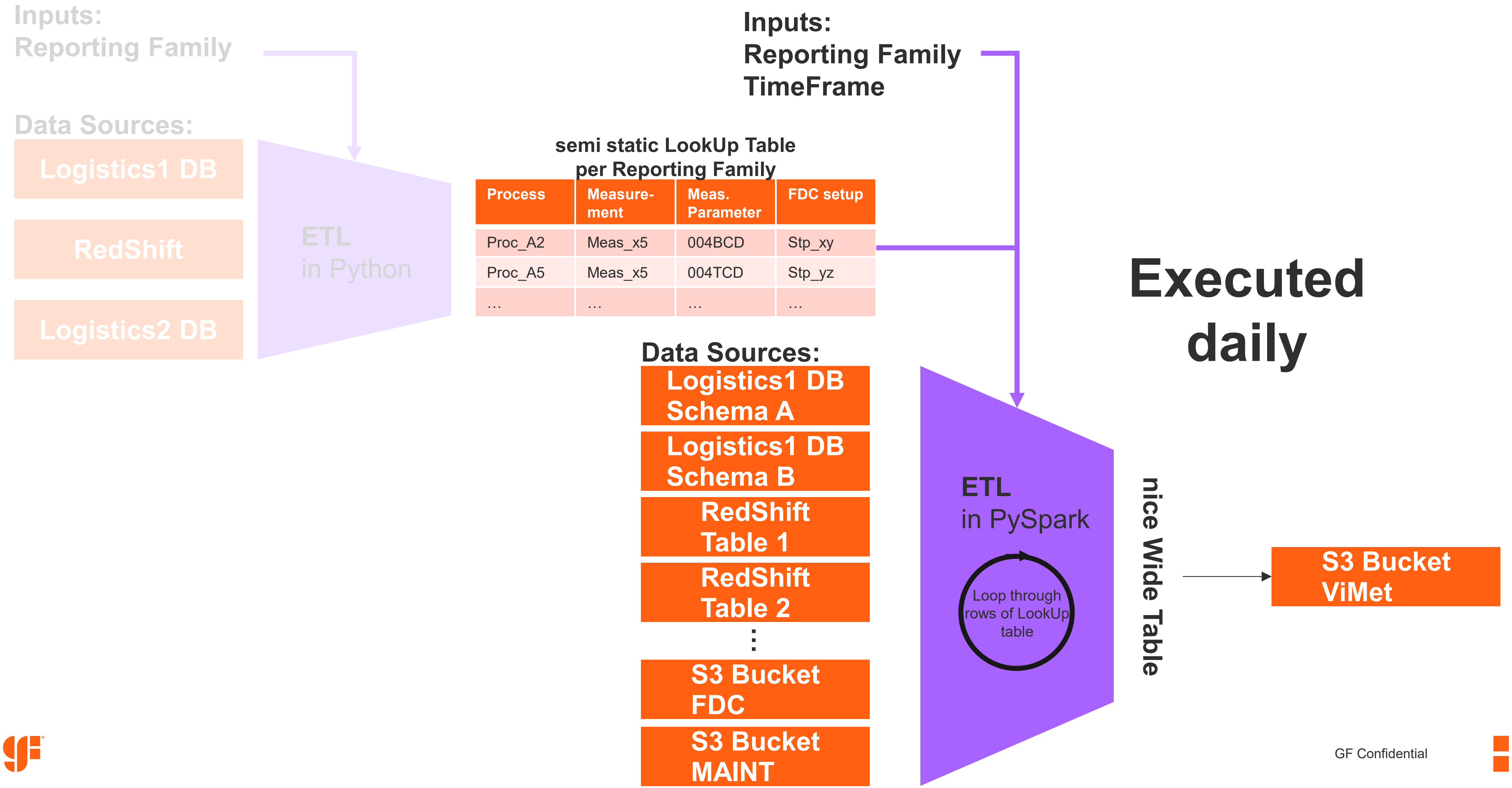
Example: Scalable Virtual Metrology



semi static LookUp Table per Reporting Family

Process	Measurement	Meas. Parameter	FDC setup
Proc_A2	Meas_x5	004BCD	Stp_xy
Proc_A5	Meas_x5	004TCD	Stp_yz
…	…	…	…

Example: Scalable Virtual Metrology



Inputs:
Reporting Family

Data Sources:
- Logistics1 DB
- RedShift
- Logistics2 DB

ETL
in Python

**semi static LookUp Table
per Reporting Family**

Process	Measure-ment	Meas. Parameter	FDC setup
Proc_A2	Meas_x5	004BCD	Stp_xy
Proc_A5	Meas_x5	004TCD	Stp_yz
…	…	…	…

**Inputs:
Reporting Family
TimeFrame**

Data Sources:
- Logistics1 DB Schema A
- Logistics1 DB Schema B
- RedShift Table 1
- RedShift Table 2
- ⋮
- S3 Bucket FDC
- S3 Bucket MAINT

ETL
in PySpark

Loop through rows of LookUp table

nice Wide Table

S3 Bucket ViMet

Executed daily

Dr. Paul Jungmann

Agenda

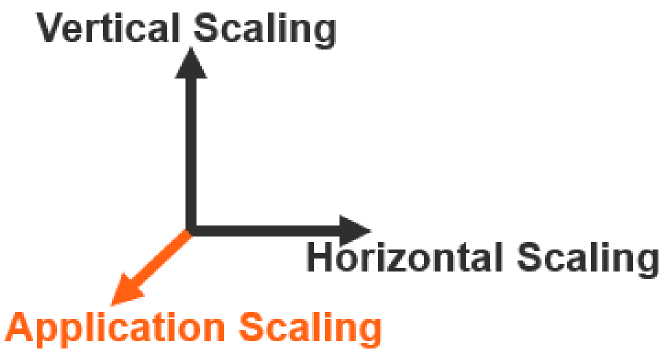
Intro to GlobalFoundries and Semiconductor Fabrication

Beyond Horizontal & Vertical Scaling

Example: Scalable Virtual Metrology

Summary

Summary



- Scaling in DataEngineering can go beyond Vertical & Horizontal

- Application scalability is often crucial for real-world cases

- Resolve dependencies dynamically when possible

- Understand which parts of the ETL pipeline must be serial and which can be parallelized

- **real-world cases mostly come with data inconsistencies**
  → understand the granularity of the problem
  → decouple static, semi static and dynamic data accesses
  → implement "control structures"

Data Engineering in Application Scaling

Specific for 1 Application

Pro	Con
Fast initial development	Redevelopment for each application
Lighter/Faster code	Hard to maintain

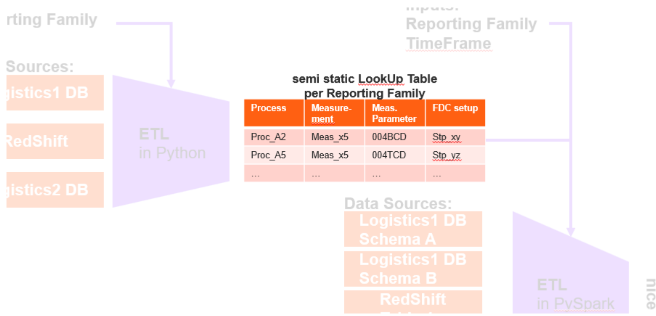
→ Usually no option

How to find the sweet spot?

Generic for all Applications

Pro	Con
Made to be reused	Hard to develop
Better Maintenance	Bloated code, special cases

→ Usually no option



semi static LookUp Table per Reporting Family

Process	Measure-ment	Meas. Parameter	FDC setup
Proc_A2	Meas_x5	004BCD	Stp_xy
Proc_A5	Meas_x5	004TCD	Stp_yz
…	…	…	…

Thank you