IEEE CEDA DATC RDF and METRICS2.1: Toward a Standard Platform for ML-Enabled EDA and IC Design

ASP-DAC 2022 Tutorial-1

January 17, 2022

Jinwook Jung, Andrew B. Kahng, Seungwon Kim,
and Ravi Varadarajan

ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE
DAC

Introduction

- Welcome to Tutorial 1 at ASP-DAC 2022 !
- Tutorial material: https://github.com/ieee-ceda-datc/aspdac-2022-tutorial
- Our speakers

Andrew B. Kahng

Seungwon Kim

Ravi Varadarajan

Jinwook Jung

Andrew B. Kahng, UC San Diego

Andrew B. Kahng is Distinguished Professor of CSE and ECE and holder of the endowed chair in high-performance computing at UC San Diego. He was visiting scientist at Cadence (1995-97) and founder/CTO at Blaze DFM (2004-06). He is coauthor of 3 books and over 500 journal and conference papers, holds 35 issued U.S. patents, and is a fellow of ACM and IEEE. He has served as general chair of DAC, ISPD and other conferences, and from 2000-2016 as international chair/co-chair of the ITRS Design and System Drivers working groups. He is currently PI of "OpenROAD" https://theopenroadproject.org/, a $15M U.S. DARPA project targeting open-source, autonomous ("no human in the loop") tools for IC implementation. He also serves as PI and director of TILOS, The Institute for Learning-enabled Optimization at Scale, which is an NSF AI Research Institute that began operations in Nov. 2021 https://tilos.ai/.

Some recent talks can be seen at his lab homepage: https://vlsicad.ucsd.edu/

Seungwon Kim, UC San Diego

Seungwon Kim received the Ph.D. degree in electrical engineering from the Ulsan National University of Science and Technology in 2019. He is currently a Postdoctoral Researcher in the VLSI CAD lab, at the University of California San Diego. His current research interests include physical design flow optimization with machine learning. He is currently participating “OpenROAD” https://theopenroadproject.org/, a U.S. DARPA project targeting open-source, autonomous (“no human in the loop”) tools for IC implementation.

Ravi Varadarajan, UC San Diego

Ravi Varadarajan is currently a Ph.D. student in the ECE department at the VLSI CAD Lab at UC San Diego. He has worked in the EDA industry for over 30 years at AT&T Bell Labs, Cadence, Tera Systems Atrenta and Synopsys. His areas of interests have been in Synthesis, Place and Route, RTL and Physical Prototyping and 3D-IC design. He holds 13 US Patents. He is also a contributor to “OpenROAD” https://theopenroadproject.org.

Jinwook Jung, IBM Research

Jinwook Jung is a Research Staff Member at IBM TJ Watson Research Center, Yorktown Heights, NY. At IBM, he works to advance design methodologies for AI hardware accelerators and high-performance microprocessors, leveraging machine learning and cloud computing.

He received Ph.D. from KAIST, Korea.

Tutorial Overview

- This tutorial is about IEEE CEDA DATC's efforts toward standard platform for ML-enabled EDA and IC design

Agenda

- **Part 1:** Introduction
  [09:00–09:10]
- **Part 2:** Challenges in ML-Enabled EDA and IC Design
  [09:10–09:35]
- **Part 3:** IEEE CEDA DATC Robust Design Flow
  [09:35–10:00]
- **Part 4:** Enabling Large-Scale Design Experiments with Cloud
  [10:00–10:40]
- Break
  [10:40–10:45]
- **Part 5:** METRICS 2.1: Toward Standard Metrics Collection System
  [10:45–11:10]
- **Part 6:** AutoTuner: Autonomous RTL-to-GDS Flow Tuning
  [11:10–11:40]
- **Part 7:** Summary and Future Work
  [11:40–12:00]








Part 2

Challenges in ML-Enabled EDA and IC Design

Andrew B. Kahng

Challenges in ML-Enabled EDA and IC Design

- Why we care
- Generic challenges
- Infrastructure challenges
- “Measure to Improve”: METRICS2.1
- True no-human-in-the-loop: AutoTuner

Learning, Optimization, Scaling

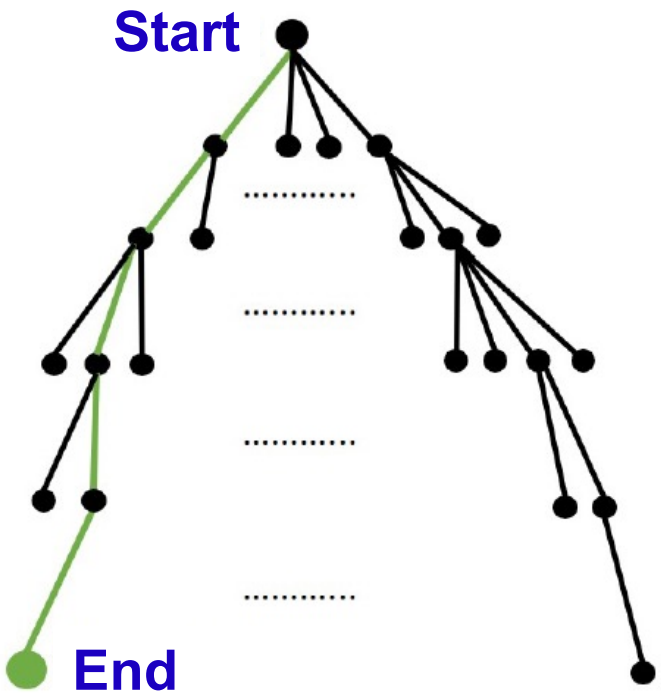
- "Machine **Learning** (ML) is the part of AI studying how computer agents can improve their perception, knowledge, thinking, or actions **based on experience or data**."

  Prof. Christopher Manning, Stanford, Sept. 2020
  https://hai.stanford.edu/sites/default/files/2020-09/AI-Definitions-HAI.pdf

- **Optimization** is the universal quest to *do better*.

- **Scaling** is what drives all of us.

Challenge: Optimization (IC Design) "Lives in a Box"

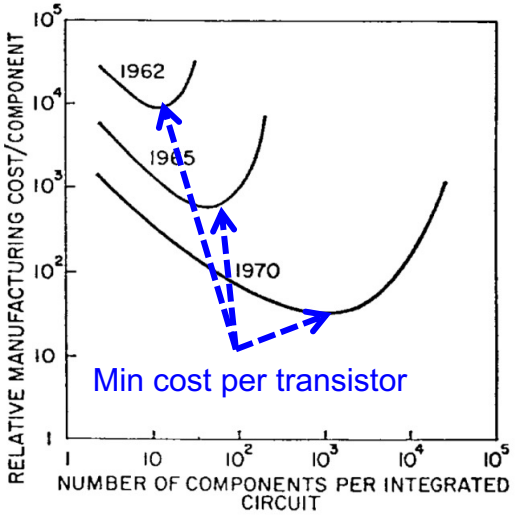


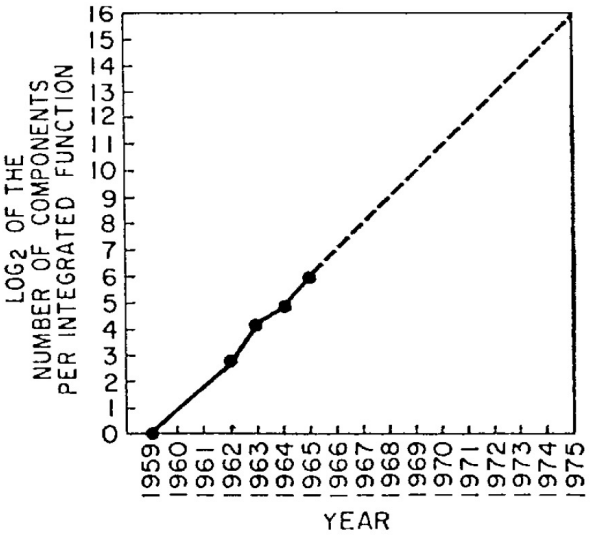
Huge space of trajectories: architecture, enablement, IPs, tools, manual fix, …

- **Start to End: expensive!**
  - O(year) for product
  - O(weeks) for SP&R and Opt
- **Goal: best possible End**
- **Constraint: stay in "Box"**
  - {compute}
  - X {licenses}
  - X {people}
  - X {weeks}

Scaling: Delivers Value

- **Moore, 1965**: "The complexity for minimum component costs has increased at a rate of roughly a factor of two per year"





- **Scaling focus: "PPAC"** power, performance, area, cost
- **Moore's Law is a law of cost reduction** 1% = 1 week
  - Corollary: greater reach of integration, more innovation within reach

ML for EDA and IC Design: What

- **Predict**
  - Will RouteOpt finish with clean signoff, <1000 DRVs by tomorrow night?
- **Classify**
  - Out of these 50 floorplans + budgets, which 3 should go into trial SP&R?
- **Estimate**
  - How many hold buffers will tool eventually add into this post-CTS layout?
- **Guide / advise**
  - What P&R tool setup/script will obtain the best QOR within next 36 hours?
- More broadly: answer any question that is difficult for humans
  - Google Brain, 2020: "super-human macro placement" on arXiv
  - **Overarching: "intelligent flow", "automated super-human expertise"**
- More directly: regressions and image classifications (LSF, litho)

ML for EDA and IC Design: Why

A. You need models to have predictions

B. You need predictions to leverage in exploration

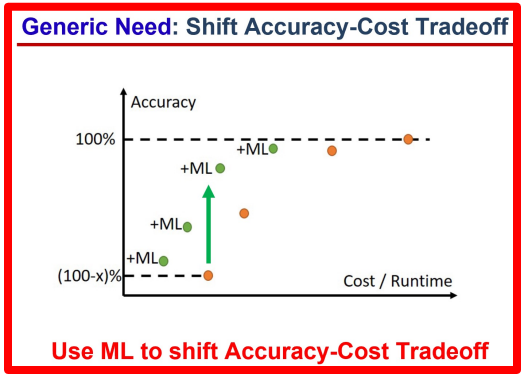
C. What you can't predict, you guardband

D. What you don't explore, you leave on the table

E. C and D are bad for product quality and schedule

- We are in an Era of Optimization
  - Look for ML to win quality, schedule, cost
  - E.g., reduce analysis runtime, miscorrelation

→ We hope that ML will bring Scaling



4 Aspects of ML for EDA and IC Design

1. Mechanization and Automation

   Create super-human robot engineers

2. Orchestration of Search and Optimization

   Optimize the use of N robot engineers

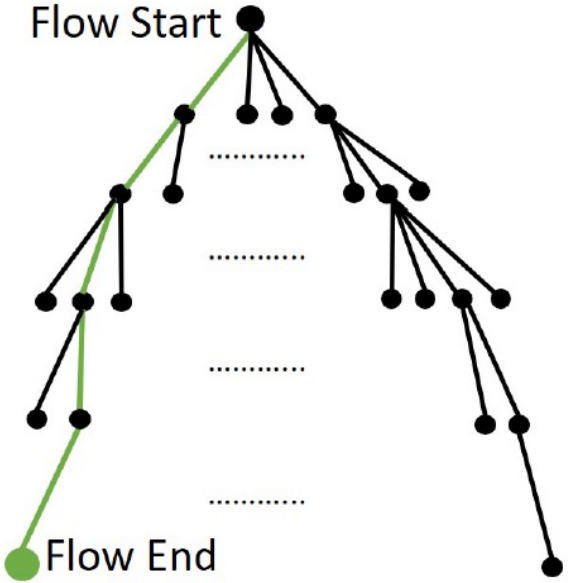
3. Pruning via Predictors, Models

   Predict design-specific tool outcomes

   Prune "doomed runs"

4. From Reinforcement Learning to Intelligence

   Target: "MLDA", "self-driving tools and flows", "superhuman"

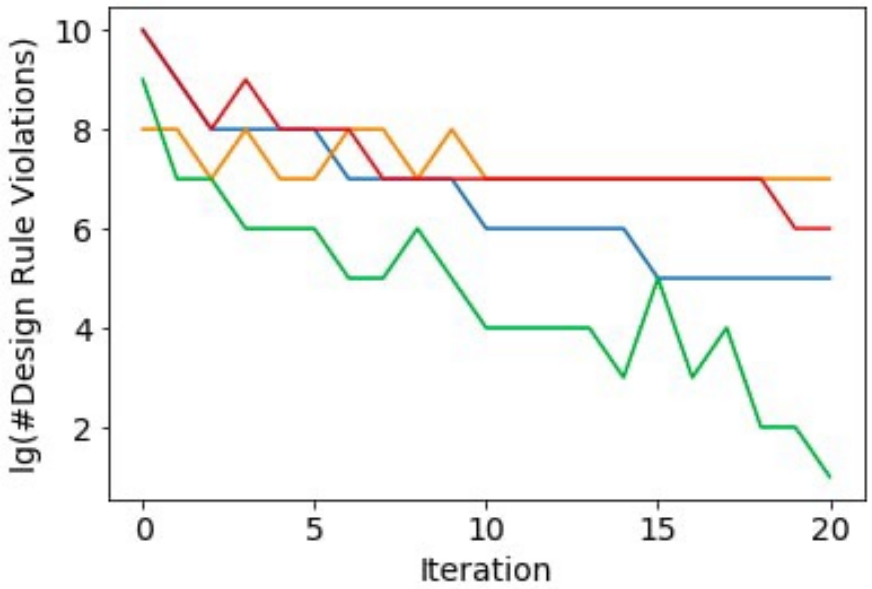


Huge space of tool, command, option trajectories through design flow

Generic Need: Predict Doomed Runs

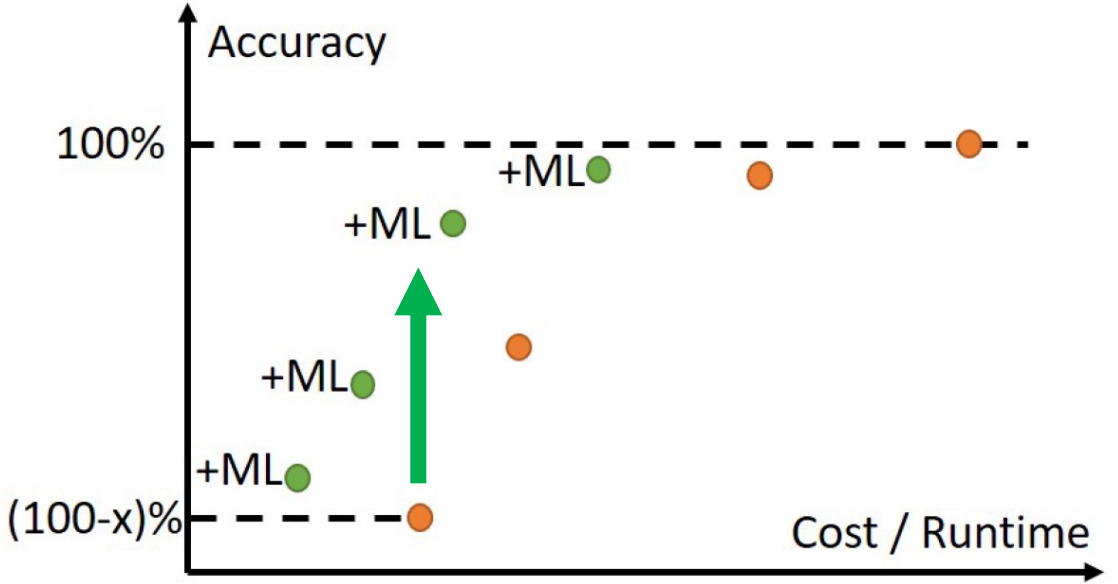
Figure from link link link

- Example: progression of DRC violations in commercial router
- Simple strategy: **track and project key metrics as time series**
- Example method: use Markov decision process (MDP): "GO" vs. "STOP" strategy card to terminate "doomed runs" early

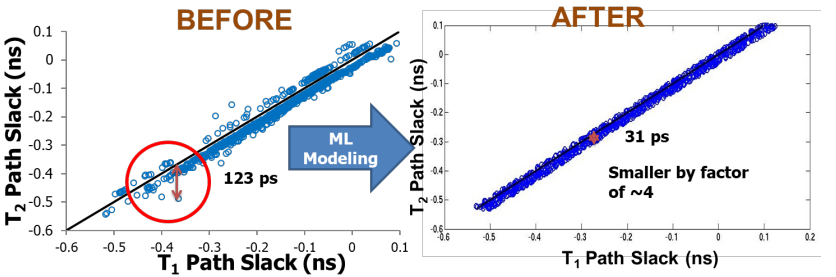


Generic Need: Shift Accuracy-Cost Tradeoff

DATE14 link SLIP15 link



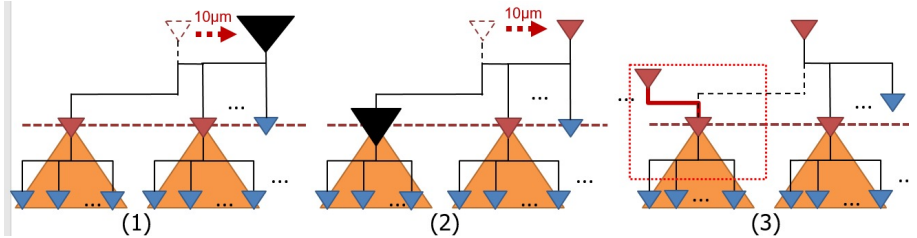
Use ML to shift Accuracy-Cost Tradeoff

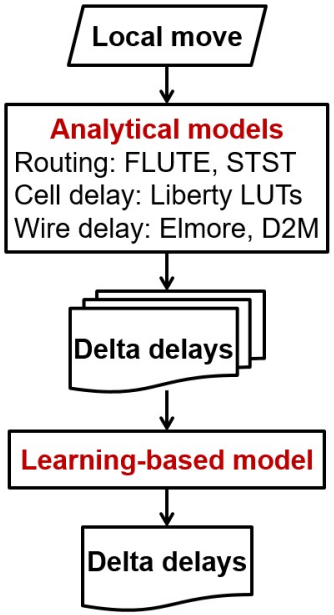


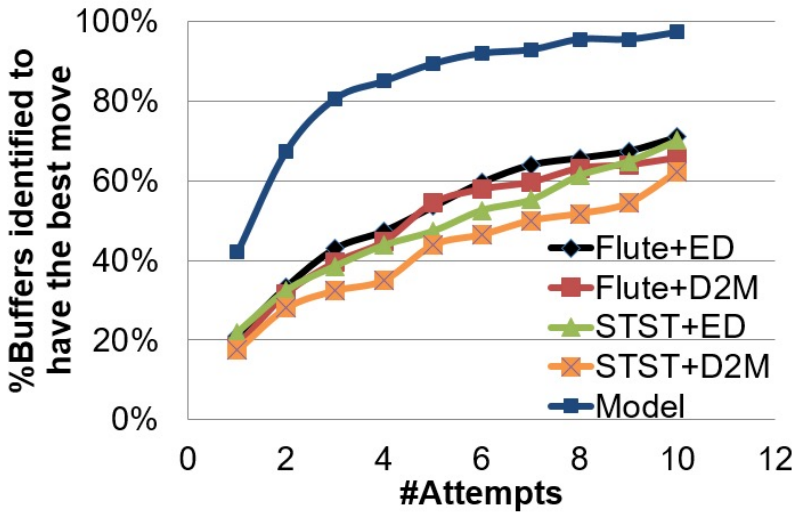
Generic Need: Model-Guided Optimization

DAC15 link

- Which CTS tweak will improve skew variation across corners?







Infrastructure Challenges and Opportunities

- **Big data and learning for IC designers**
  - Model encapsulation and application
  - IP preservation to enable model sharing
- **Big data and learning for EDA tools/flows**
  - Data model, names, semantics **including derived data**
  - Enable design-specific **modeling and prediction** of tool/flow outcomes
    Recall: http://vlsicad.ucsd.edu/GSRC/metrics
- **Machine learning requires data**
  - **Real designs, Artificial designs and "Eyecharts"**
  - **Calibration data and more** ("Underwriters Lab")
  - **Develop training data together** – grid computing paradigm
    - **Year 2000**: SETI@Home **Year 2020**: Tool X on PDK Y with IP Z ?
  - + Challenges and incentives: "Kaggle for ML in IC design"
  - IEEE CEDA DA Technical Committee: "Metrics4ML" https://github.com/ieee-ceda-datc/datc-rdf-Metrics4ML

Infrastructure Principles

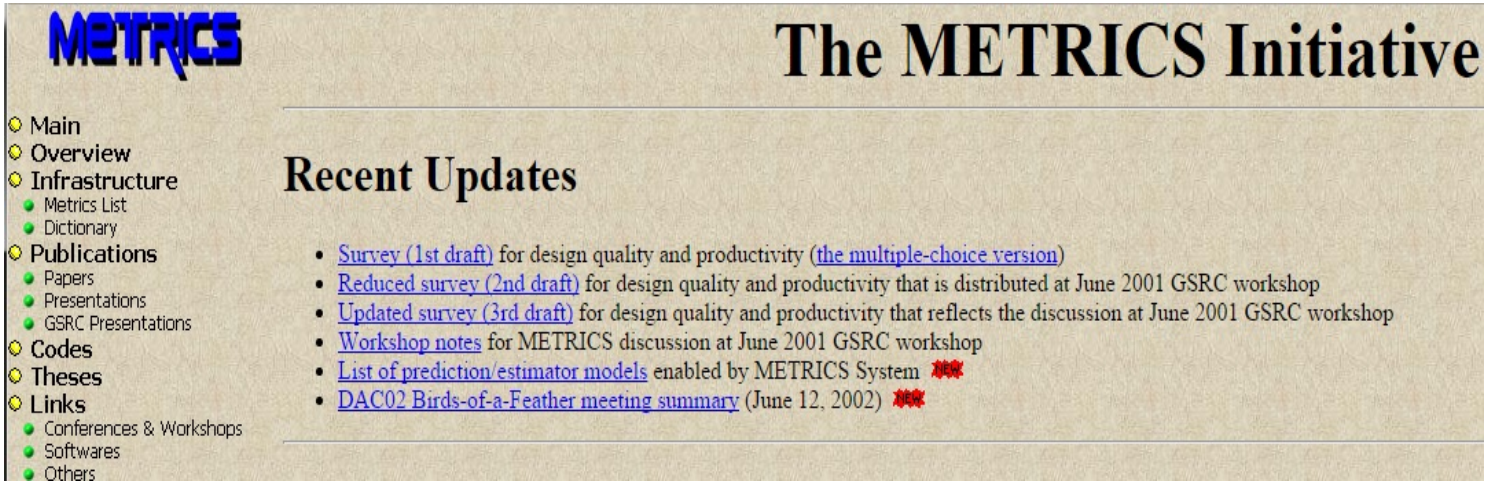
- General and extensible
  - Syntax and semantics to support future-proofing – e.g., integration and physics
  - 3D/2.5D, reliability, thermal/stress, …
- Open, transparent and unencumbered
  - Many landmines: copyrighted command names, reporting formats, “de facto standards” → staying aware and clear of these helps solve the Tower of Babel problem!
  - **Permissive** open sourcing is best: BSD/MIT/Apache licenses
- No ambiguity!
  - METRICS2.1 discussed later today: each measurement concept maps to a **unique** METRICS2.1 metric name
  - Each METRICS2.1 metric name maps to a **unique** interpretation as a measurement
  - **1-1 correspondence is crucial to avoid future confusion**
- Free, open and frictionless to everyone; agnostic to EDA vendor

ML in IC Design Requires a Standard API!

- **Support for ML in IC design**
  - **Standards** for model encapsulation, model application, and IP preservation when models are shared
- **Standard ML platform for EDA modeling**
  - Enable design metrics collection, tool/flow model generation, design-adaptive tool/flow configuration, prediction of tool/flow outcomes
- **Datasets to support ML**
  - Real designs and artificial designs
  - Shared training data – e.g., analysis correlation, post-route DRV prediction, optimal sizing
- Challenges and incentives: “Kaggle for ML in IC design”

"METRICS"

[DAC00, ISQED01]

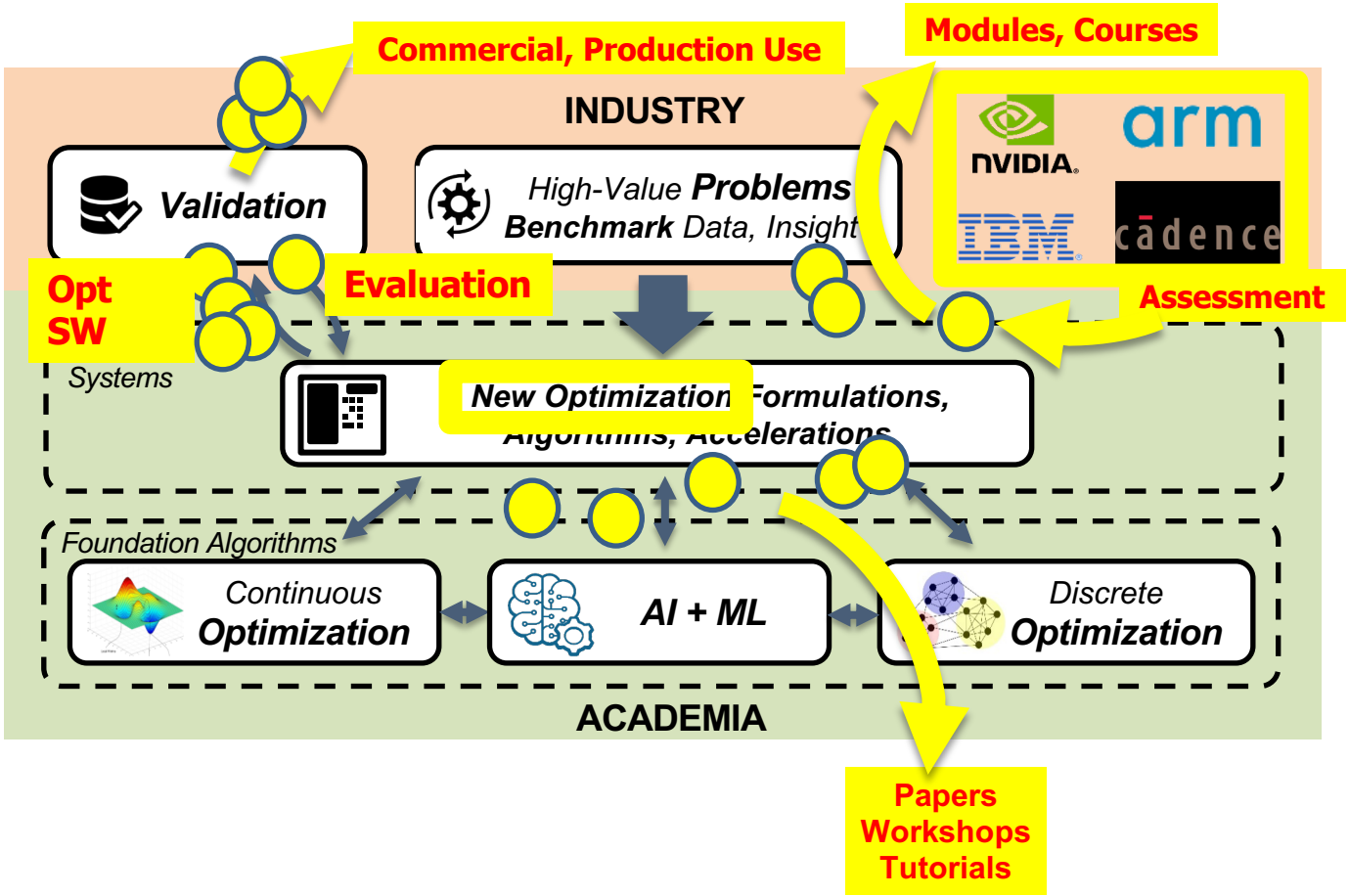


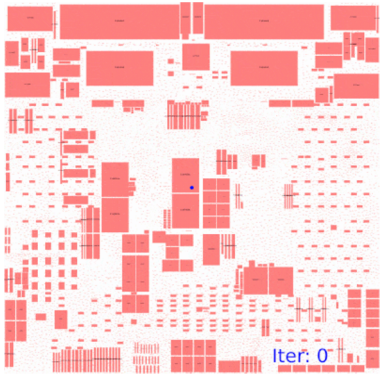
- **METRICS 1.0** (1999; DAC00, ISQED01)
  - **"Measure to Improve"** http://vlsicad.ucsd.edu/GSRC/metrics
- **METRICS 2.0** (WOSET-2018) was proposed as an update of **METRICS 1.0**
- **METRICS2.1** is proposed as an extension to **METRICS 2.0**

Flow Hyperparameter AutoTuner

- A "No-Human-in-Loop" for RTL-to-GDS flow using the **OpenROAD** tool
- Automatic iterative tuning for QoR improvement within a given hyperparameter range space
- Interface: Python packages Ray/Tune
- Search algorithms: HyperOpt, PBT, Optuna, Nevergrad, Ax, random search
- Advantages:
  - Pre-acquired big data is not necessary for execution.
  - Less experimental trials compared to random or grid parameter sweeping
  - Powerful parallelization management → core/thread management, external server usage, web visualization
  - Suboptimality gap reduces significantly – and according to available CPU resource

Lifecycle of Learning + Optimization + Impact in Practice





DREAMPlace: UT+Nvidia

(Discrete-Continuous)

THANK YOU !

- Research at UCSD is supported by NSF, DARPA, Qualcomm, Samsung, NXP, Mentor Graphics and the C-DEN center.

- Questions/Feedback: abk@eng.ucsd.edu

Part 3

IEEE CEDA DATC Robust Design Flow

Jinwook Jung

ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE
DAC

Part 3: Outline

- Introduction
  - IEEE CEDA DATC
  - DATC Robust Design Flow (RDF)
- The DATC RDF flow
  - Chisel and hardware generator designs
  - RTL obfuscation with ASSURE
  - Logic synthesis with Yosys+ABC
  - DFT insertion with Fault
  - P&R with OpenROAD or point tool-based flow
- Using DATC RDF: A short demo

Design Automation Technical Committee (DATC)

- **A technical committee of IEEE Council on EDA (CEDA)**
  - Provide a forum for discussing strategies and issues in design automation
- **Tasks**:
  1. Academic reference design flow, *DATC Robust Design (RDF)* → **Part 3**
  2. Standard platform for ML-enabled EDA → **Parts 4-6**

Committee	Affiliation	Role
Jinwook Jung	IBM Research, USA	Chair / Integration
Andrew B. Kahng	UC San Diego, USA	Co-chair / Flow
Victor N. Kravets	IBM Research, USA	Logic synthesis
Jianli Chen	Fudan Univ., China	Placement
Yih-Lang Li	National Yang Ming Chiao Tung Univ., Taiwan	Routing
Iris Hui-Ru Jiang	National Taiwan Univ., Taiwan	Previous chair

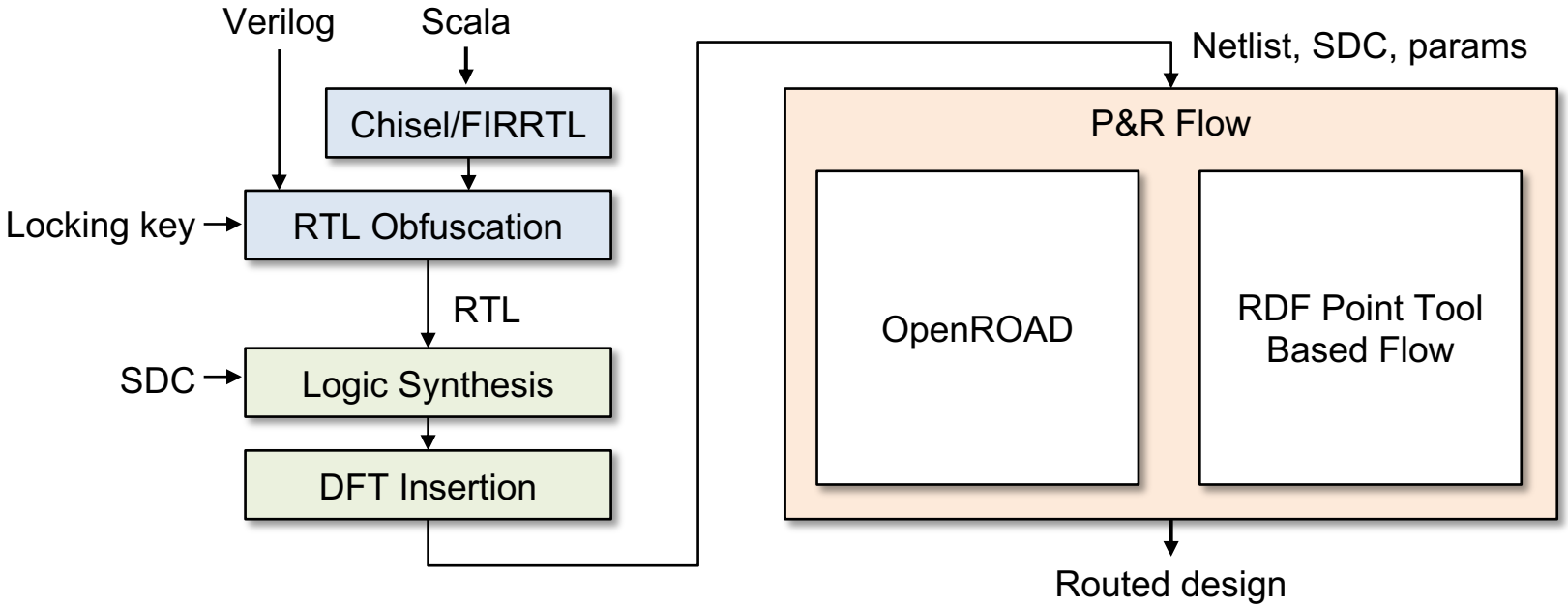
DATC Robust Design Flow (RDF)

- **Academic reference design flow**
  - Initiated 2016; latest release 2021
  - Design flow built upon academic tools
  - Also supports complete, industrial format-based RTL-to-GDS flow with OpenROAD
- **GitHub link:**
  - https://github.com/ieee-ceda-datc/datc-rdf
- **Goal:**
  1. Preserve leading research codes, and integrate them into design flow
  2. Trigger design flow-scale and cross-stage optimization research

Component	Tool
RTL generator	Chisel/FIRRTL
RTL obfuscation	ASSURE
Logic synthesis	Yosys+ABC
DFT insertion	Fault
Floorplanning	TritonFP
Global placement	RePlAce, FZUplace, NTUPlace3, ComPLx, Eh?Placer, FastPlace3-GP
Detailed placement	OpenDP, MCHL, FastPlace3-DP
Flip-flop clustering	Mean-shift, FlopTray
Clock tree synthesis	TritonCTS
Global routing	FastRoute4-lefdef, NCTUgr, CUGR
Detailed routing	TritonRoute, NCTUdr, DrCU
Layout finishing	KLayout, Magic
Gate sizing	Resizer, TritonSizer
Parasitic extraction	OpenRCX
STA	OpenSTA, iTimerC
Integrated P&R app	OpenROAD
Database	OpenDB
Libraries/PDK	NanGate45, SKY130, ASAP7, NCTUcell
Benchmark conversion	RosettaStone

DATC RDF: Flow Overview

- Starts from **pure Verilog** or **Chisel-generated Verilog** (optionally obfuscated)
- **Logic synthesis** followed by **DFT insertion**
- P&R with **OpenROAD** or **point tool-based RDF flow**



Chisel and Hardware Generator Designs

CHISEL
https://www.chisel-lang.org

- **Chisel: C**onstructing **H**ardware **I**n **S**cala **E**mbedded **L**anguage
  - Provides special classes, objects, language convention for hardware design
  - Easy to implement highly-parameterized, abstract ***hardware generators***

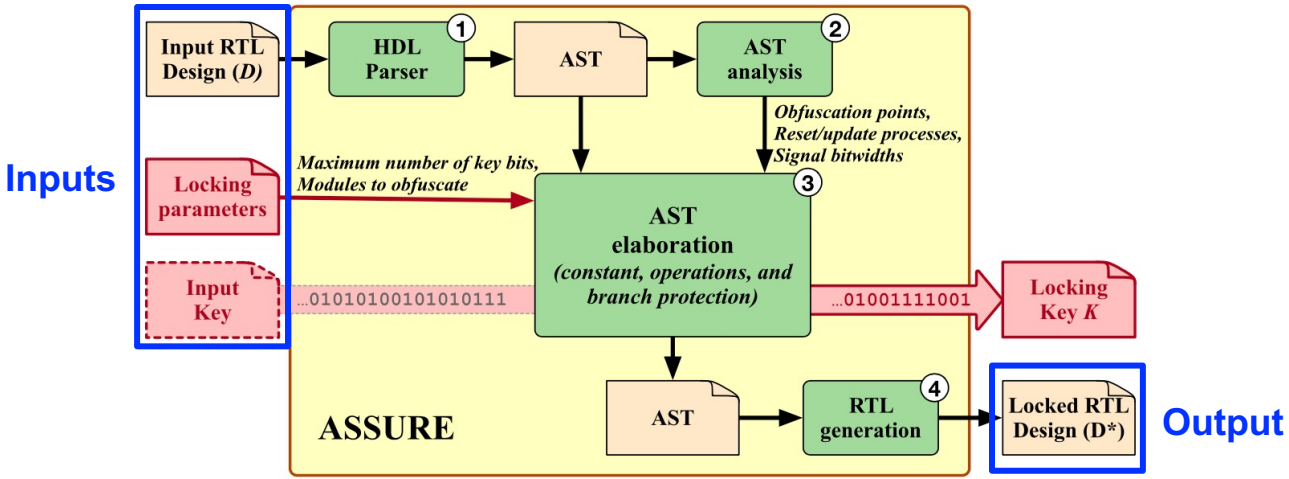
- **Substantial interest from hardware architecture researchers**
  - RISC-V Rocket Chip Generator
  - Gemmini deep learning accelerator
  - Hwacha vector processor

- With Chisel, we can **use recent processor/accelerator designs in design flow research**

```
import chisel3._

class Filter (bitWidth: Int) extends Module {
  val io = IO(new Bundle {
    val in = Input(UInt(bitWidth.W))
    val out = Output(UInt(bitWidth.W))
  })
  val z1 = RegNext(io.in)
  val z2 = RegNext(z1)
  io.out := (io.in*1.U) + (z1*1.U) + (z2*1.U)
}

// Parameterized instantiation
val movingSum3Filter = Module(new Filter(8))
```

RTL Obfuscation with ASSURE

- **ASSURE:** RTL obfuscation framework assuming netlist-only threat model
- **3 obfuscation techniques supported:**
  1. **Constant**: substitute constants in RTL with secret locking key
  2. **Operation**: adds redundant operators alongside original operators in RTL
  3. **Branch**: obfuscates branch conditions in RTL



[Pilato et al., IEEE TVLSI 2021]

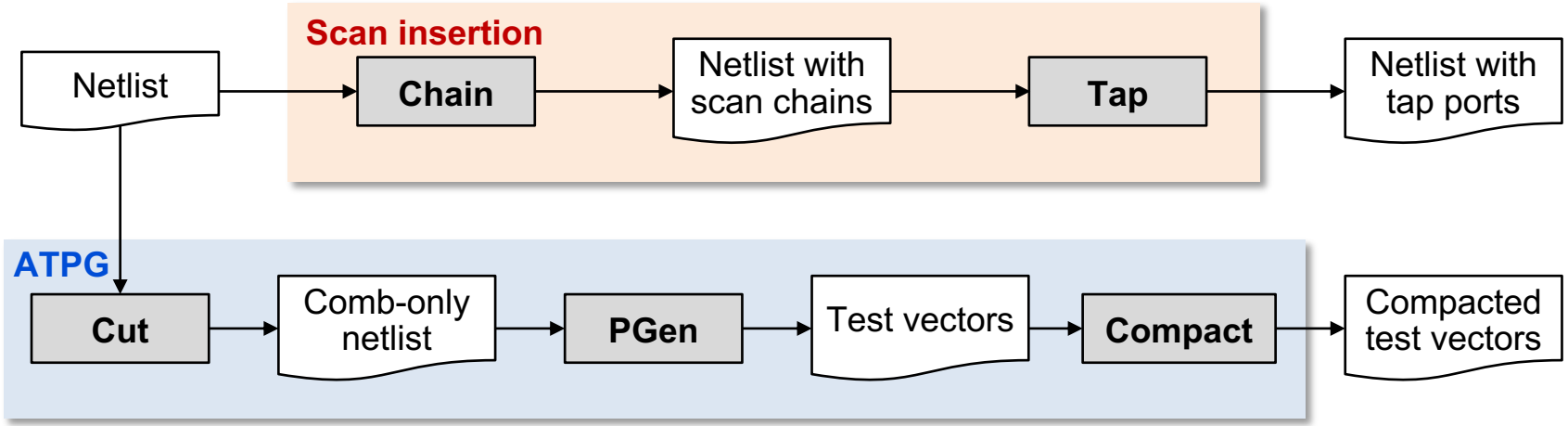
Logic Synthesis using Yosys+ABC

- **Logic synthesis in RDF is done with Yosys+ABC**
  - GitHub link: https://github.com/The-OpenROAD-Project/yosys
- **Latest ABC updates incorporated in DATC RDF**
  - Tight integration of SAT solver with internal design representation
  - Improved scalability and new options to improve technology mapping quality
- **LazyMan synthesis scripts:**
  - Reuse logic rewriting choices empirically discovered in earlier runs
  - Reduce runtime while improving quality

```
abc 01> &cec -x nn.170.aig <-------- New option "-x"
cnn.187.aig Networks are equivalent. Time = 1238.07 sec
abc 02> &cec cnn.170.aig cnn.187.aig
Networks are equivalent. Time = 12058.28 sec
```

DFT Support with Fault

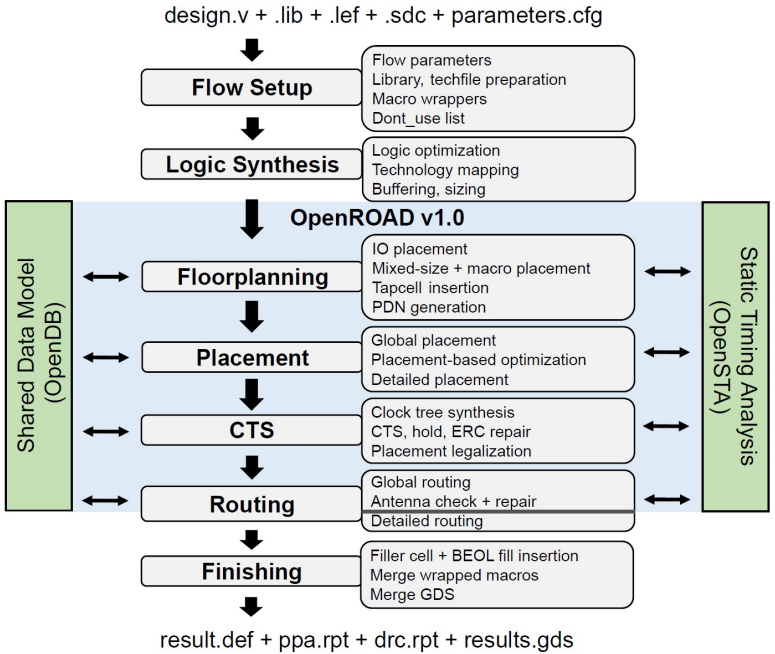
- **Fault:** open-source DFT infrastructure
  - GitHub link: https://github.com/Cloud-V/Fault
  - **Scan chain insertion:** add test signals, scan chains, and JTAG interface
  - **ATPG:** generate compressed test vectors



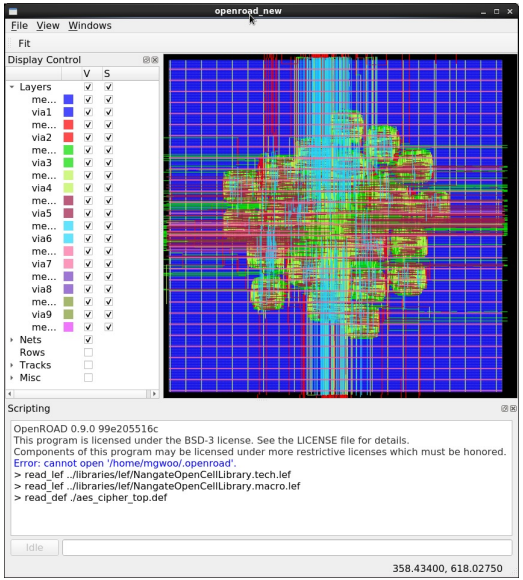
P&R Flow: OpenROAD

OpenROAD
https://theopenroadproject.org

- **OpenROAD:** famous open-source tool for complete RTL-to-GDSII flow
  - GitHub link: https://github.com/The-OpenROAD-Project
  - Enable robust handling of industry standard design enablements in RDF



OpenROAD GUI



P&R Flow: Point-Tool Based Flow

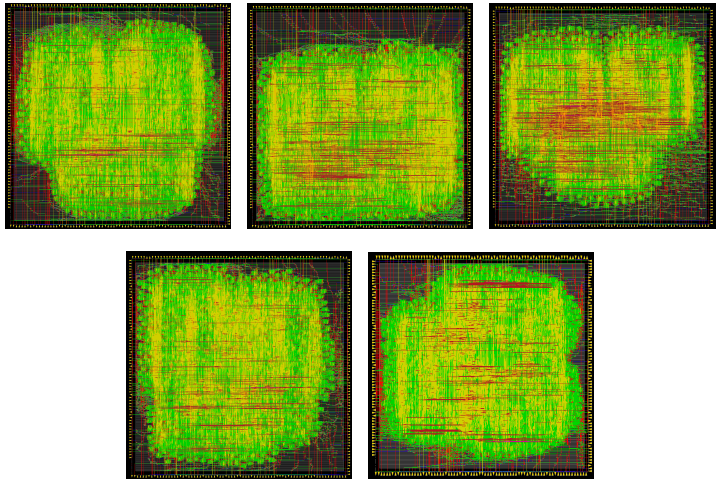
- **Composed with academic point tools**, especially contest-winning tools
  - GitHub link: https://github.com/ieee-ceda-datc/RDF-2020
  - Preserve leading research codes and integrate them into complete design flow

Flow definition YAML

```
flow:
  - stage: floorplan
    tool:  TritonFP
  - stage: global-place
    tool:  FZUplace
  - stage: detail-place
    tool:  FastPlace-3
  - stage: cts
    tool:  TritonCTS
  - stage: global-route
    tool:  FastRoute4-lefdef
  - stage: detail-route
    tool:  TritonRoute
```

.v, .sdc, .lib, .lef

DATC RDF Point Tool Based Flow



Design outcomes

Using DATC RDF

- **DATC RDF consists of flow collateral and tool binaries**
  - Flow collateral in GitHub: https://github.com/ieee-ceda-datc/datc-rdf
  - Tool binaries provided in Docker image: https://hub.docker.com/r/jinwookjung/rdf

- **Installing and launching RDF:**

```
# Clone DATC RDF repository
git clone https://github.com/ieee-ceda-datc/datc-rdf.git
cd datc-rdf
git submodule update --init

# Launch DATC RDF
docker run --rm -it \
    -v $(pwd)/submodules/OpenROAD-flow-scripts/flow:/openroad-flow \
    -v $(pwd)/example:/root/example \
    jinwookjung/rdf
```

Demonstration of DATC RDF

1. RTL generation with Chisel
2. RTL obfuscation with ASSURE
3. Logic synthesis with Yosys
4. DFT insertion with Fault
5. P&R with OpenROAD

Summary

- **DATC RDF:** Academic reference **RTL-to-GDS** design flow
  1. Preserve/integrate leading research codes in complete design flow
  2. Trigger design flow and cross-stage optimization research

- **The DATC RDF Flow**
  - Starts from pure Verilog or Chisel-generated Verilog (optionally obfuscated)
  - Logic synthesis followed by DFT insertion
  - P&R done with OpenROAD integrated app or point tool-based RDF flow

- **GitHub link:** https://github.com/ieee-ceda-datc/datc-rdf

Part 4

Enabling Large-Scale Design Experiments with Cloud

Jinwook Jung

ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE

Outline

- Introduction and overview
- Docker and Kubernetes
- Kubernetes cluster architecture for DATC RDF
- Scaling-out design experiments
  - Multi-container design experiments using Ray
  - Multi-node design experiments using using Ray and public cloud services
- Summary

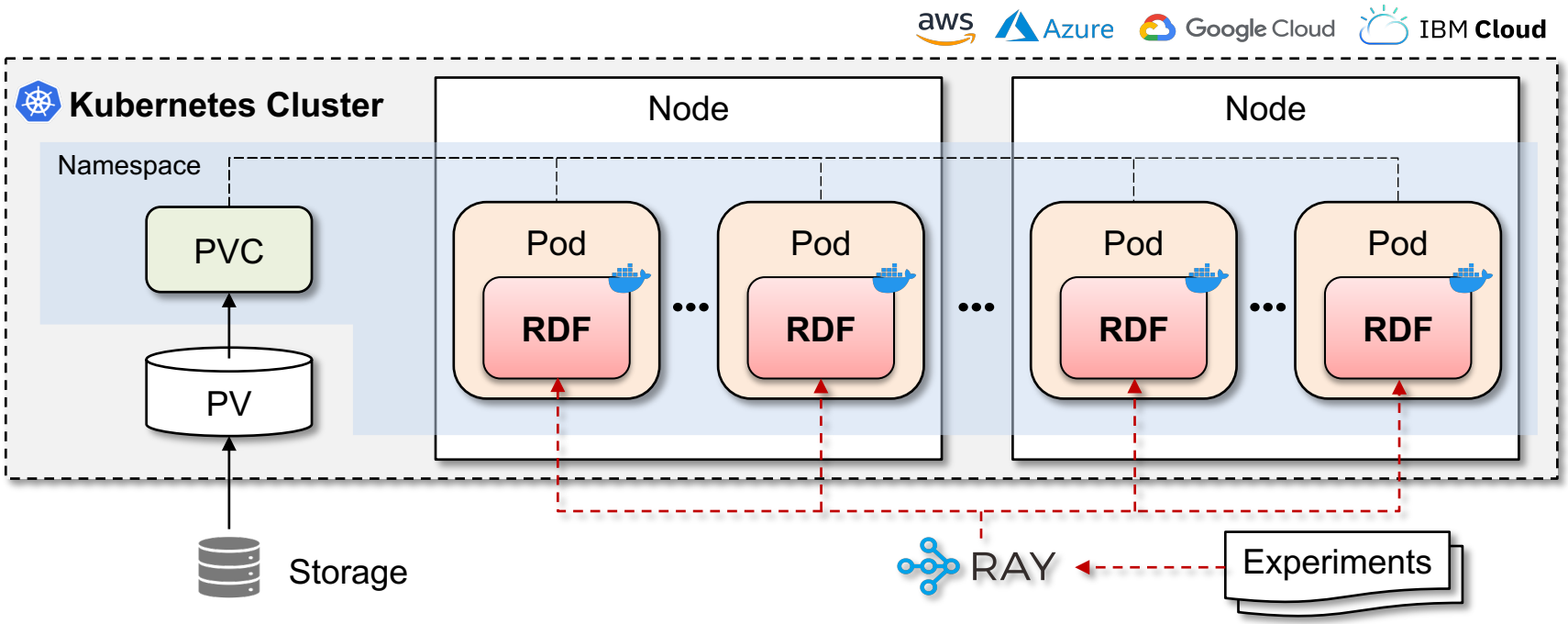
Introduction

- **"*Machine learning requires data*"**
  - Generating data for ML-enabled EDA needs compute resource and time
  - Efficient way to generate large amount data for ML-enabled EDA?

- **Public cloud provides compute resources on-demand**
  - How to best-utilize public cloud for ML-CAD?
  - Obtain cloud instance, install libraries, compile tools, ... → Not scale well
  - Need to establish ***generic* and *cloud-native* way** for enabling large-scale design experiments on public cloud services

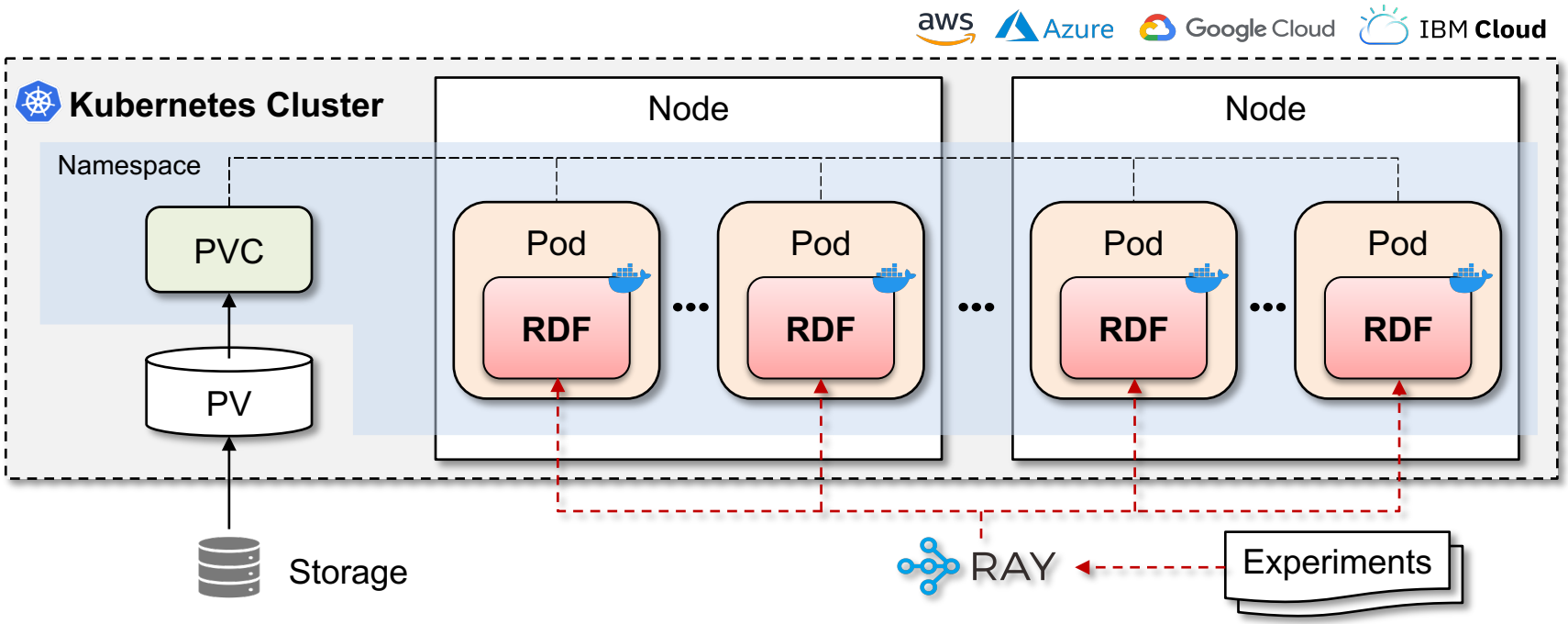
Goal of Part 4

- We will learn how to deploy large-scale design experiments on cloud, utilizing *Docker, Kubernetes*, *Ray*, and *public cloud services*



Overview of Part 4

1. Basics of Docker and Kubernetes
2. Minimal Kubernetes cluster for design experiments
3. Scale-out to multi-container, multi-node cluster using public cloud services



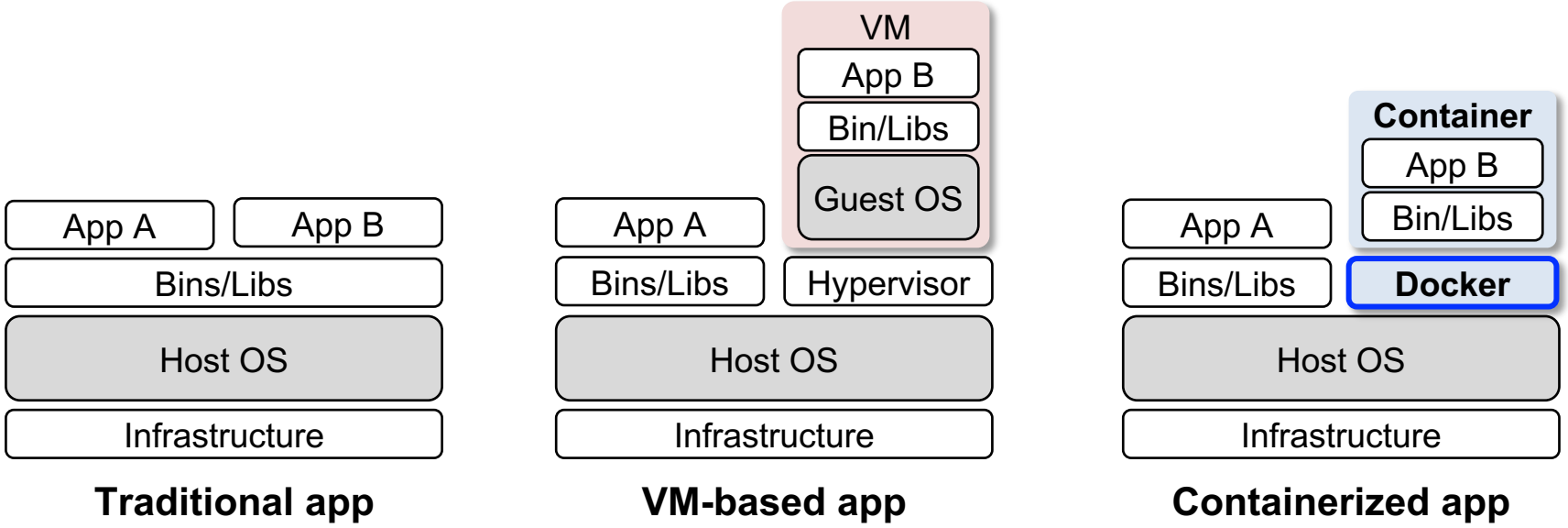
Outline

- Introduction and overview
- Docker and Kubernetes
- Kubernetes cluster architecture for DATC RDF
- Scaling-out design experiments
  - Multi-container design experiments using Ray
  - Multi-node design experiments using using Ray and public cloud services
- Summary

Docker and Container

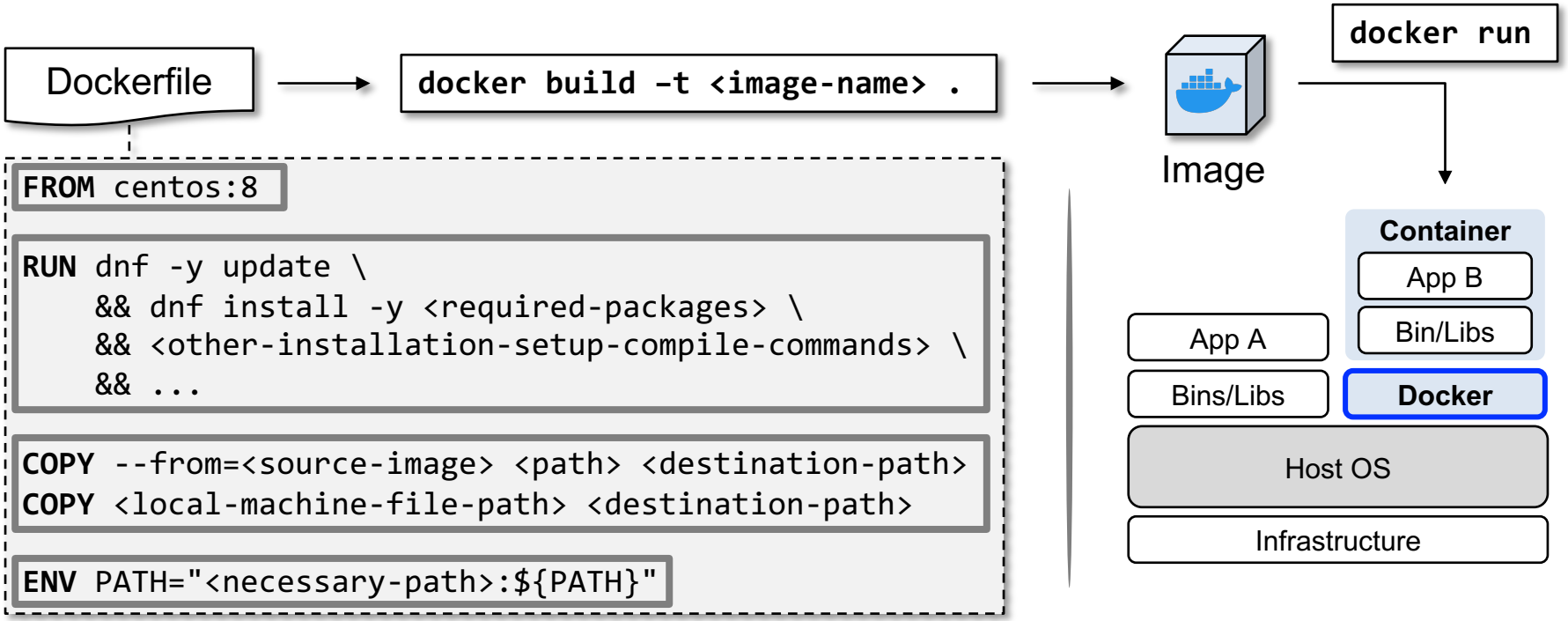
- **Docker:** creates application into *container*
  - **Container** = application + dependencies (libs/bins)
  - Isolate applications from each other, while sharing same OS kernel
- Comparison to different types of applications

https://docker.com



Docker Image and Runtime

- **Image:** application packaged by Docker and deployable as container
- **Docker runtime:** executes container from image

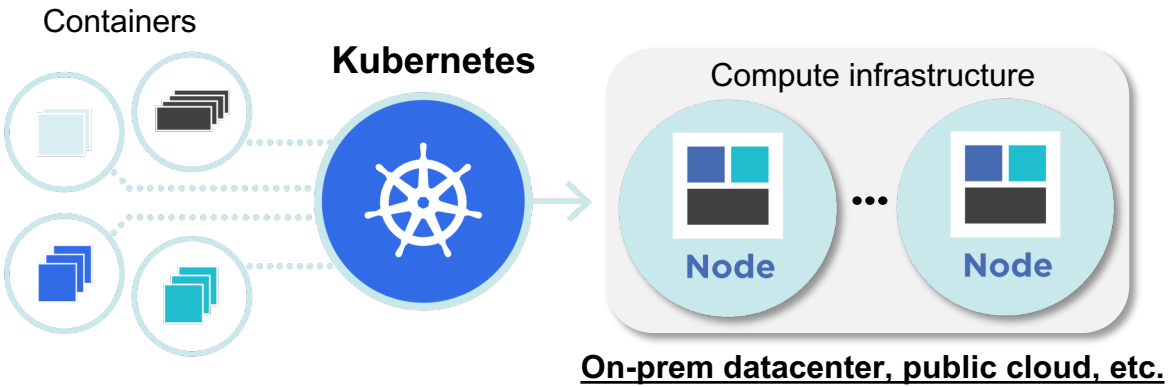


Kubernetes and Container Orchestration

- **Docker:** build and run containers
- **Kubernetes:** provide *orchestration services*
  - Deploy, scale, and manage containers on large-scale compute infrastructure
  - Abstracts underlying compute infrastructure (on-premise datacenter machines, local servers, and public cloud services)
  - Infrastructure agnostic → standard way to deploy containerized applications

kubernetes

https://kubernetes.io



[Image source: https://kubernetes.io]

Basic Kubernetes Terminologies

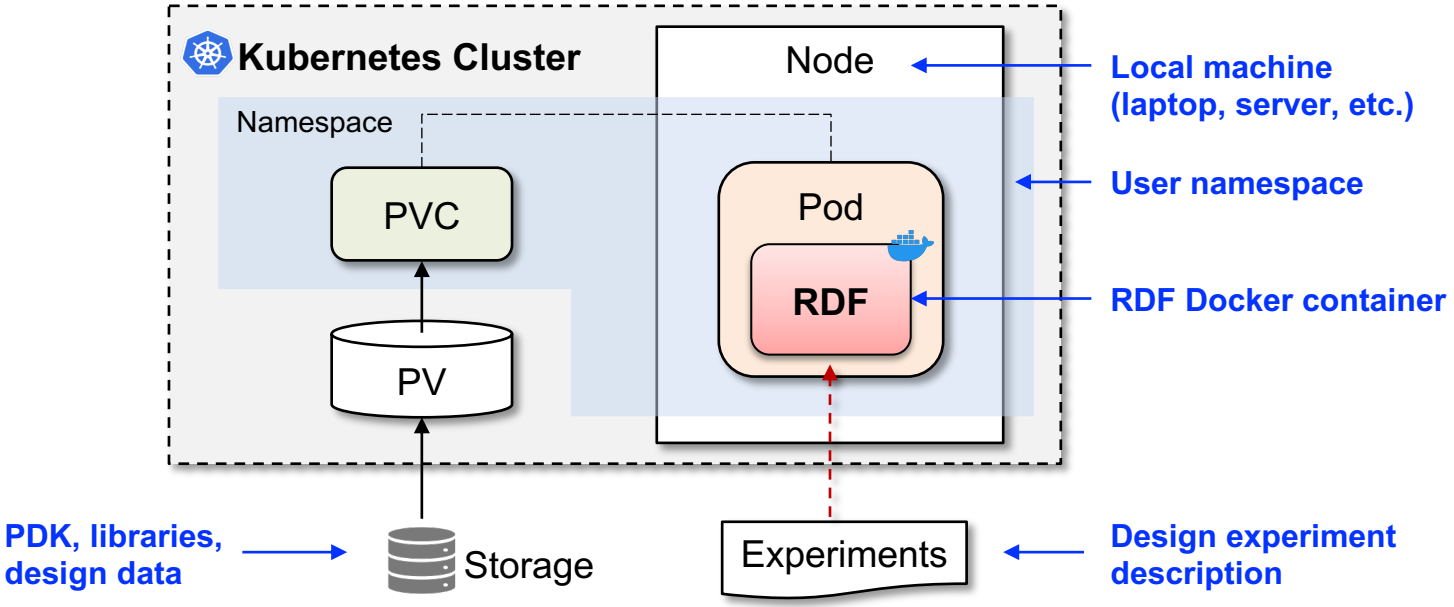
- **Node:** Compute machine, e.g., physical server, VM, cloud instance, etc.
- **Cluster:** A set of nodes
- **Pod:** Construct for running containers in Node
- **Persistent Volume (PV):** External storages made available in Cluster, such as local directory, NFS and cloud file storage
- **Persistent Volume claim (PVC):** “Ticket” authorizing Pod to mount PV

Outline

- Introduction and overview
- Docker and Kubernetes
- Kubernetes cluster architecture for DATC RDF
- Scaling-out design experiments
  - Multi-container design experiments using Ray
  - Multi-node design experiments using using Ray and public cloud services
- Summary

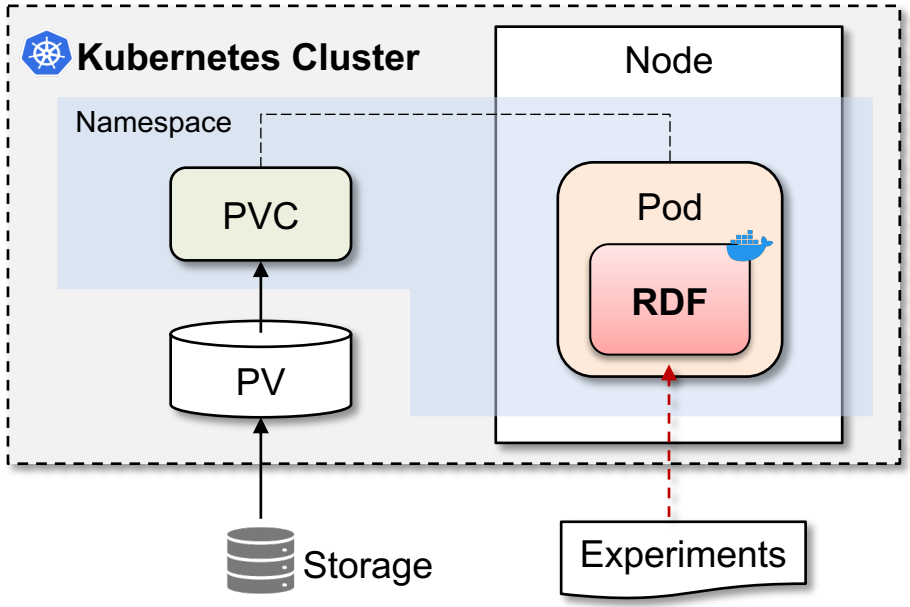
“Minimal” Kubernetes Cluster Architecture for DATC RDF

- Single node
- Single volume (PV/PVC)
- Single pod and container



Demonstration

- How to create the minimal Kubernetes cluster and run DATC RDF there

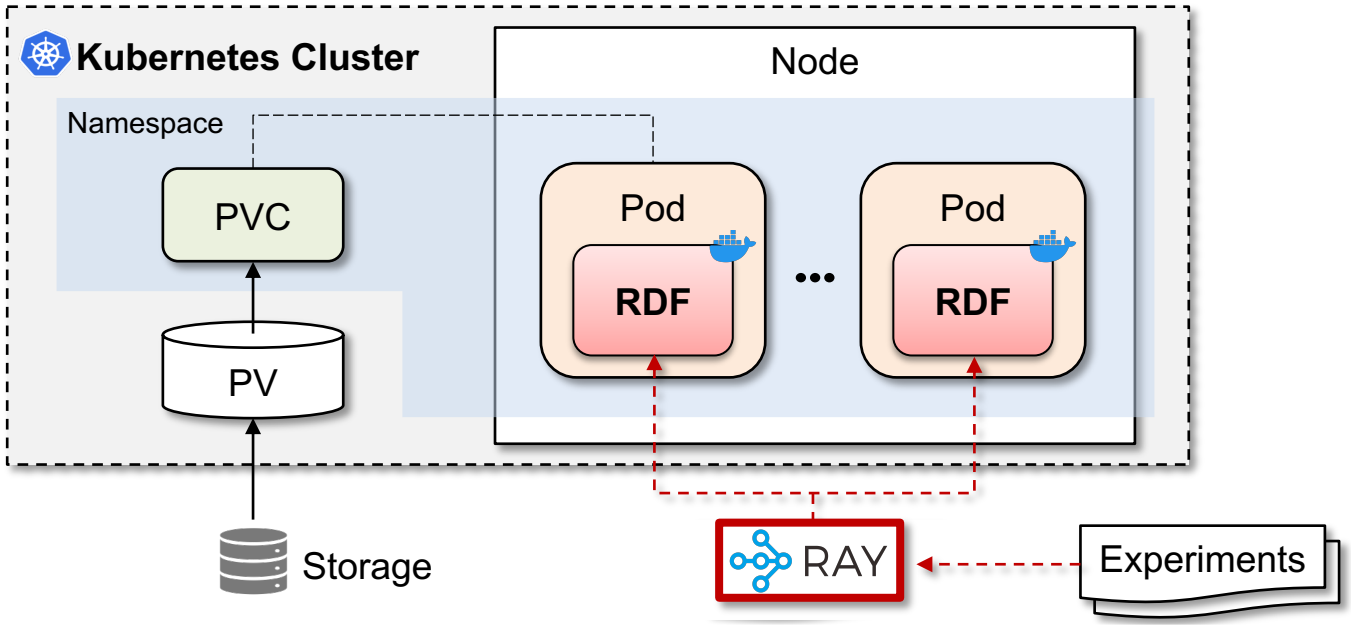


Outline

- Introduction and overview
- Docker and Kubernetes
- Kubernetes cluster architecture for DATC RDF
- Scaling-out design experiments
  - Multi-container design experiments using Ray
  - Multi-node design experiments using using Ray and public cloud services
- Summary

Scaling Out DATC RDF Containers

- Single-node/single-pod cluster → limited compute resource
- For large-scale design experiments, we want larger cluster
  - One way to achieve it is to scale-out containers/pods in Kubernetes cluster
  - How to efficiently **distribute experiments**? → We introduce ***Ray***



Ray

- **Distributed execution framework** developed by UC Berkeley
  - GitHub Link: https://github.com/ray-project/ray
  - Provides a simple Python API to build distributed applications
  - Works with Kubernetes and other cloud-based clusters

- **Example:**

Original program

```
def f(x):
    return x+1

[f(i) for i in range(100)]
```

Distributed version with Ray

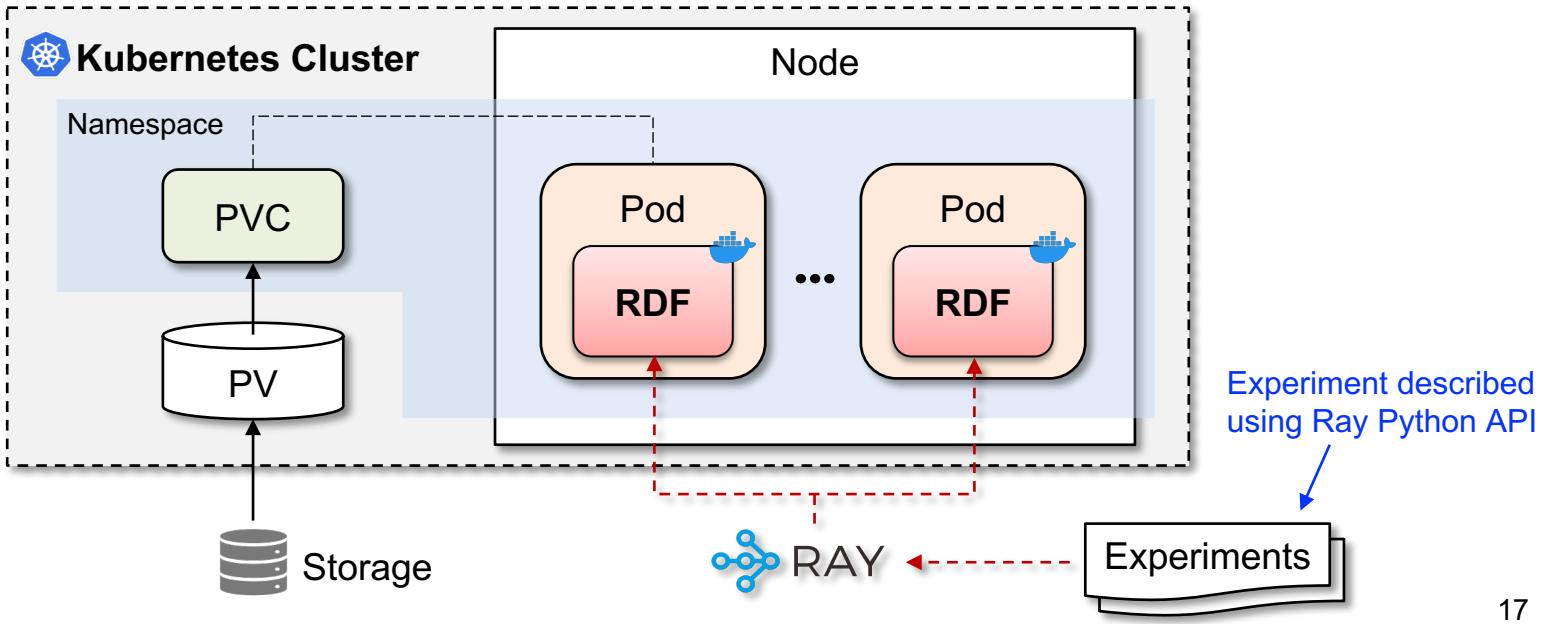
```
import ray
ray.init(...)

@ray.remote
def f(x):
    return x * x

[f.remote(i) for i in range(100)]
```

Initialize Ray

Decorate function

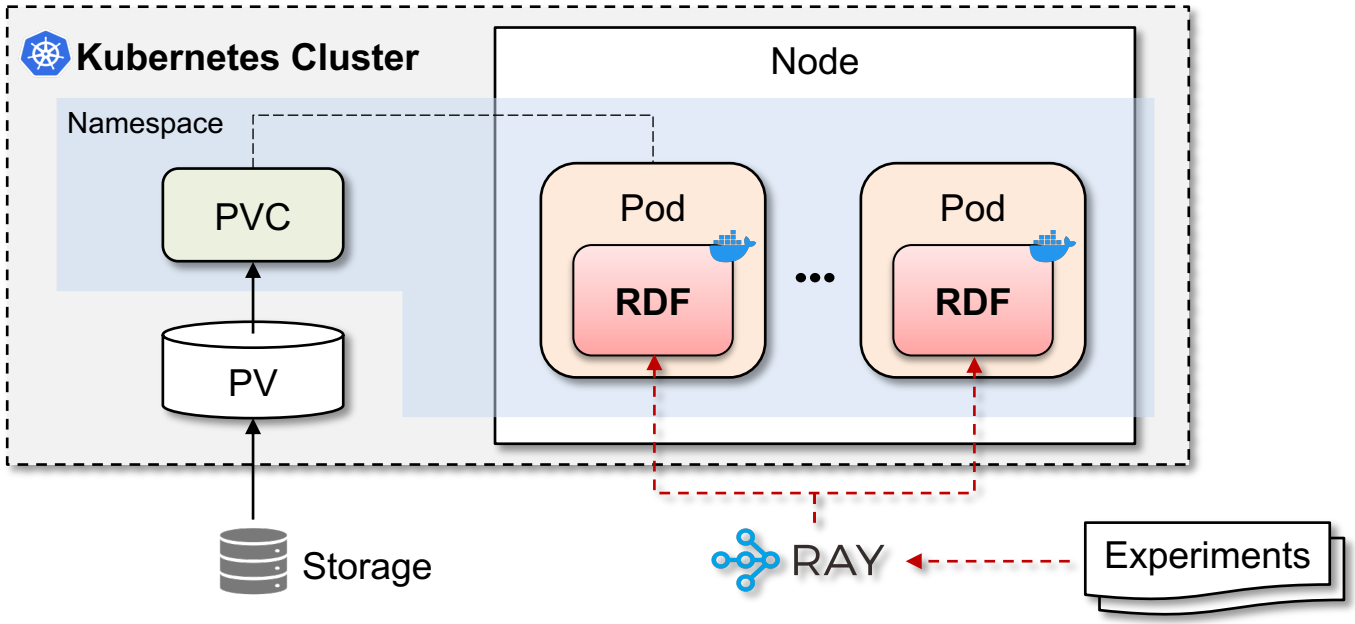
Distributed Design Experiments with Ray and Kubernetes

1. Create multi-container Kubernetes cluster
2. Describe large-scale design experiments using Ray Python API
3. Deploy experiments on Kubernetes cluster via Ray



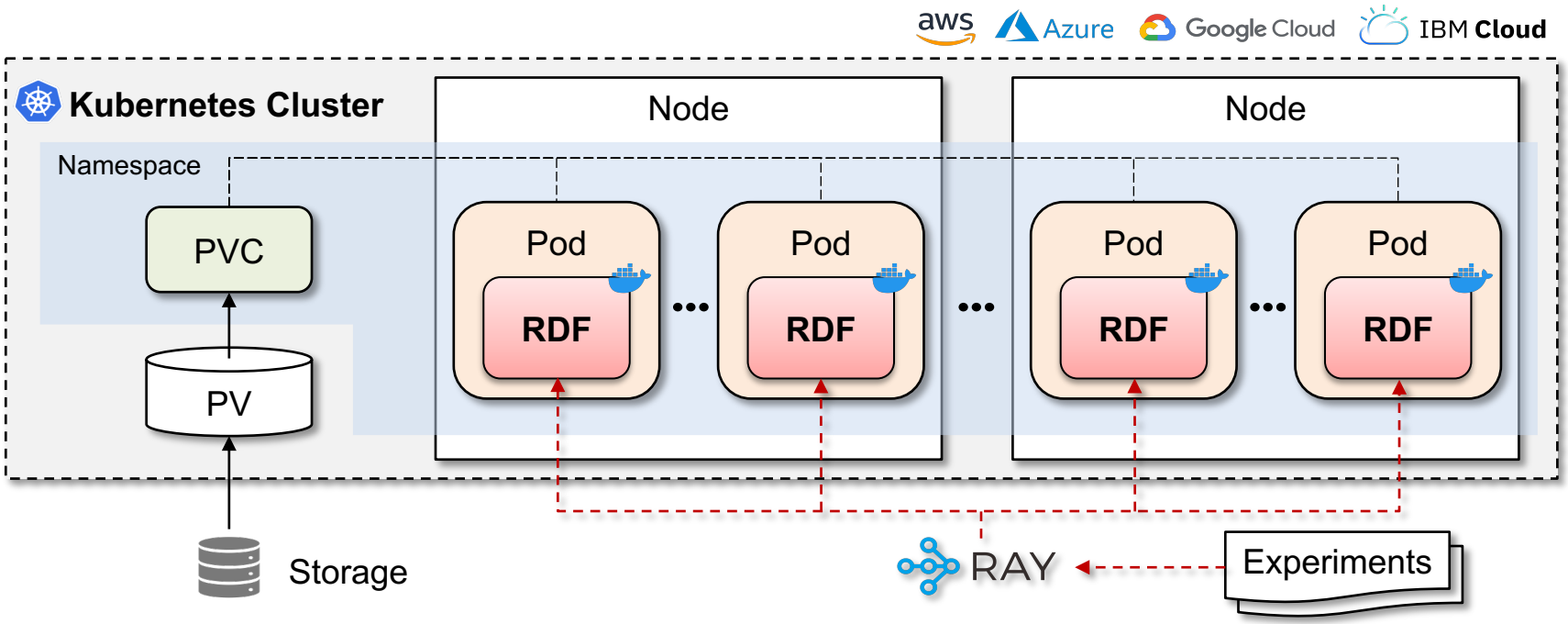
Demonstration

- Consider design experiment to find maximum achievable utilization
  - Given design, PDK, libraries
  - Try many floorplan utilizations, and check whether flow fails or not
- Use Ray and Kubernetes to distribute such experiments efficiently



Scale Out DATC RDF Containers using Public Cloud

- Multi-container Kubernetes cluster still uses *single node*
- Now, we compose **multi-node Kubernetes cluster** using public cloud service to enable even larger-scale design experiments



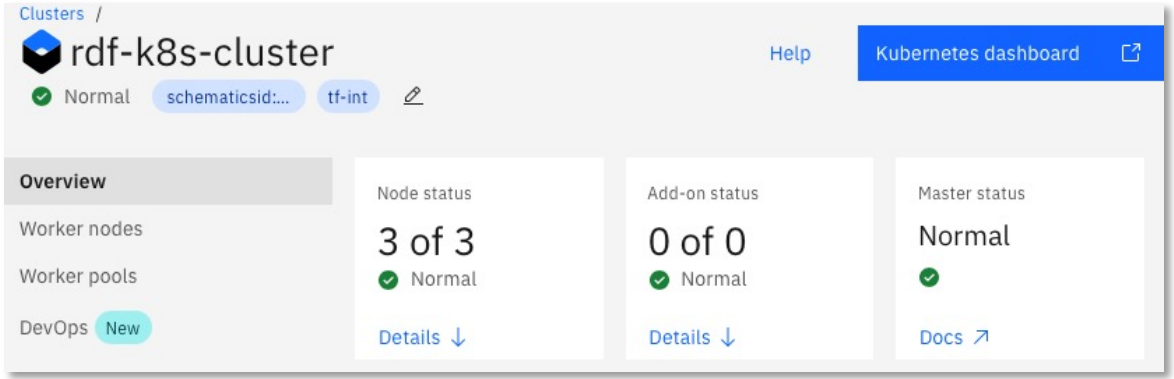
Kubernetes Available on Public Cloud

- Major public cloud providers all offer Kubernetes service
  - **AWS**: Amazon Elastic Kubernetes Service (EKS)
  - **Microsoft Azure**: Azure Kubernetes Service (AKS)
  - **Google Cloud Platform**: Google Kubernetes Engine (GKE)
  - **IBM Cloud**: IBM Cloud Kubernetes Service (IKS)

- Using public cloud, we can create multi-node Kubernetes cluster in minutes with a few clicks

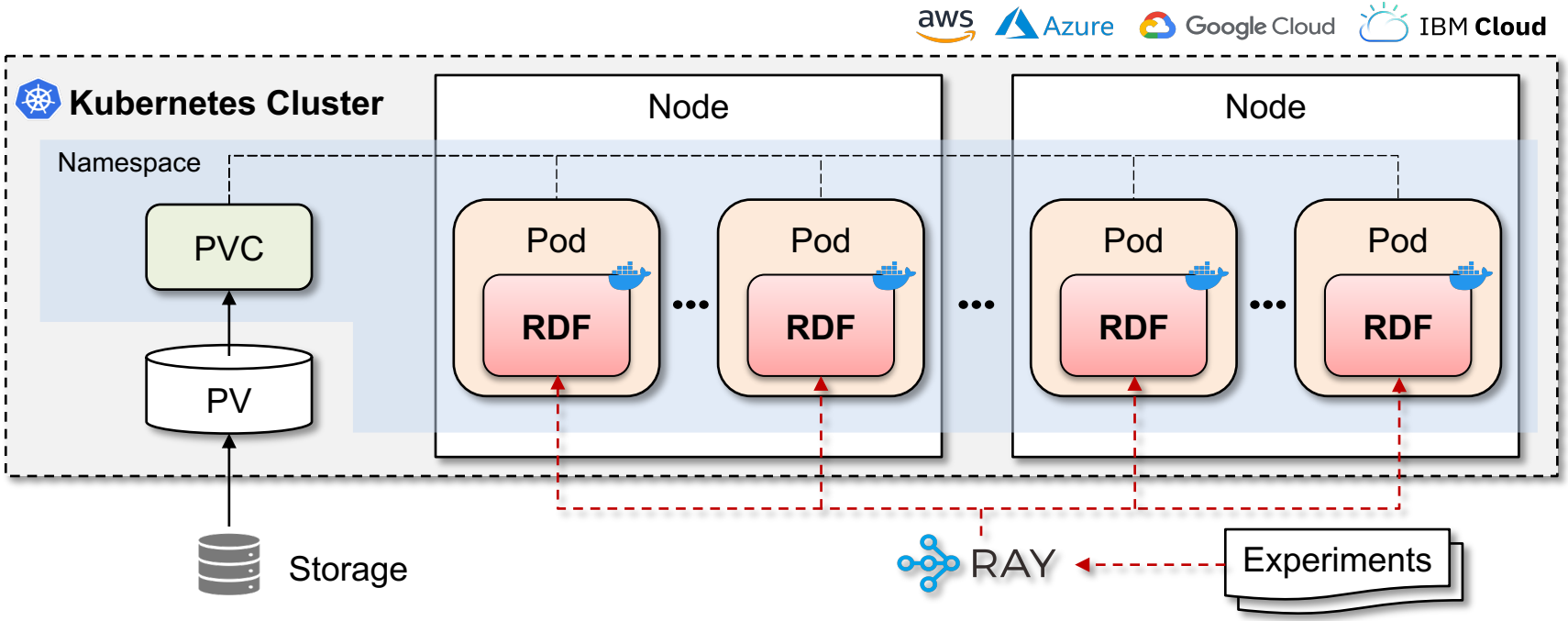
Example: Kubernetes Cluster on IBM Cloud

1. Go to https://cloud.ibm.com and create account
2. Create “Kubernetes Service”
3. Choose desired node type, i.e., CPUs and RAM, and create cluster
4. The cluster will be ready in a few minutes



Demonstration

- Once created, Kubernetes cluster on public cloud can be used in same way
- Next demo shows how to deploy distributed design experiments on public cloud using Ray and Kubernetes



Outline

- Introduction and overview
- Docker and Kubernetes
- Kubernetes cluster architecture for design experiments with RDF
- Scaling-out design experiments
  - Multi-container design experiments using Ray
  - Multi-node design experiments using using Ray and public cloud services
- Summary

Summary

- "***Machine learning requires data***" → Time and compute resources
  - We learned how to utilize public cloud to generate data for ML-CAD
  - We learned how to deploy large-scale experiments on public cloud using Ray and Kubernetes, in *generic/cloud-native way*

- **Code, notebooks**: https://github.com/ieee-ceda-datc/aspdac-2022-tutorial

- In this part, metrics collection from data were done in ad-hoc manners
  - "*Can we collect key metrics in a common, standard form?*"
  - **METRICS 2.1**: standardized metrics collection/design process recording → **Part 5**

Part 5

METRICS 2.1: Toward Standard Metrics Collection System

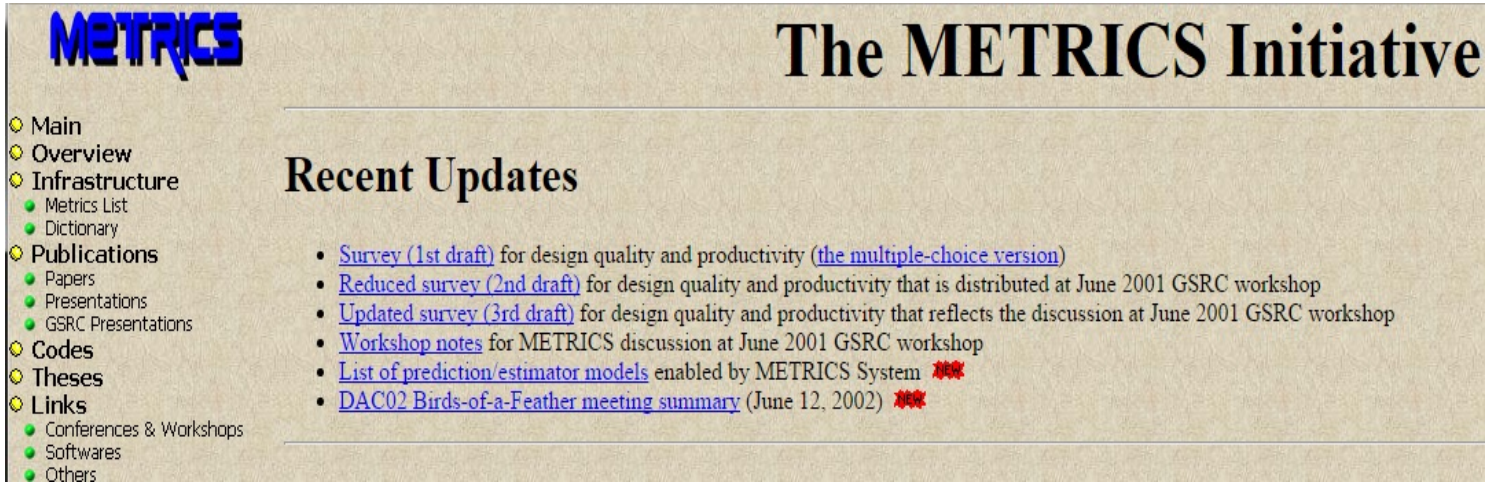
Ravi Varadarajan

ML in EDA and IC Design Requires Data and Standards

- **Standard ML platform(s) for EDA and design process modeling**
  - Enable design metrics collection, tool/flow model generation, design-adaptive tool/flow configuration, prediction of tool/flow outcomes

- **Datasets to support ML**
  - Real and artificial designs
  - Shared training
  - Challenges and incentives: “Kaggle for ML in IC design”

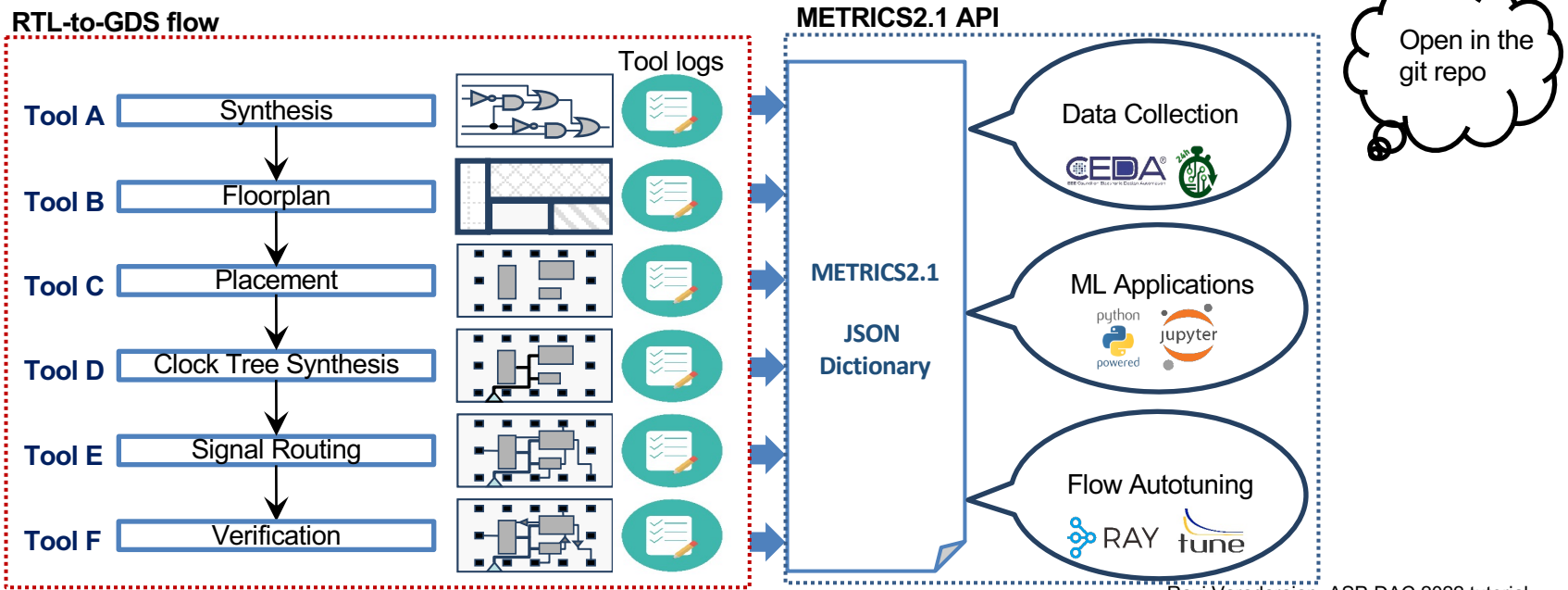
“METRICS” (DAC00, ISQED01)



- **METRICS 1.0** (1999; DAC00, ISQED01)
  - **“Measure to Improve”** http://vlsicad.ucsd.edu/GSRC/metrics
- **METRICS 2.0** (WOSET-2018) was proposed as an update of **METRICS 1.0**
- **METRICS2.1** is proposed as an extension to **METRICS 2.0**

METRICS2.1 Overview

- An initiative of the **IEEE CEDA Design Automation Technical Committee (DATC)**
- Collection and sharing of design of experiment (DoE) public datasets using the **OpenROAD** tool, with intelligent setting of constraints
- Sample Jupyter notebooks on the datasets to visualize and build simple prediction models
- True “No-Human-in-Loop”: Automatic hyperparameter tuning of engine parameters

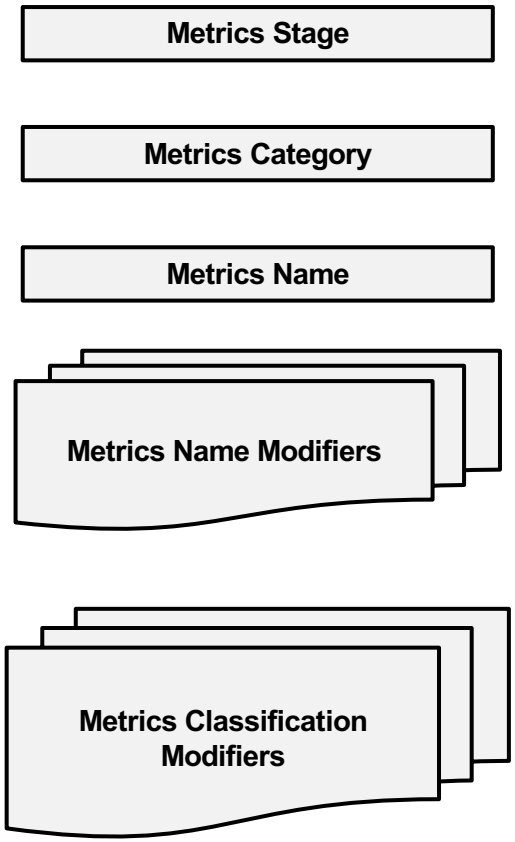


METRICS2.1 Principles

- General and extensible
  - Syntax and semantics to support future addition of new metrics
- No ambiguity!
  - Any desired measurement must map to a unique METRICS2.1 metric
  - Every METRICS2.1 metric must map to a unique interpretation as a measurement
- This two-way mapping is crucial to avoid future confusion
- Ability to capture the same metric at different stages of the design flow
- Free, open and frictionless to everyone; agnostic to any EDA vendor

METRICS2.1 Organization

- METRICS2.1 is organized as a hierarchical JSON object
- Top level: predefined *stage* or a user-defined *snapshot*
  - Default stages *are init, synth, floorplan, globalplace, detailedplace, cts, globalroute, detailedroute and finish*
- Inside each stage: metrics are further organized as a predefined metrics *category,* a metrics *name,* an optional metrics *name modifier,* and optional metrics *classification modifiers*
  - Predefined metrics categories are *flow*, *design*, *timing*, *power* and *route*
- A metric name and its optional *name mo*difier uniquely define the metric and its units, e.g., *design__instance__count*
- A metric *classification modifier* subdivides the metric into a specific type (stdcells, macros) or a specific structure (clock domain, analysis view etc.)
- Precedence order is part of the METRICS2.1 specification
- METRICS2.1 naming convention: https://github.com/ieee-ceda-datc/datc-rdf-Metrics4ML#metrics-naming-convention



METRICS2.1 Examples

- Sample metrics

Metric	Description
timing__setup__wns	Setup worst negative slack in the design
timing__setup__wns__clock:clk_a	Setup worst negative slack for clock "clk_a" in the design
timing__setup__wns__analysis_view:slow	Setup worst negative slack for the analysis view "slow"
power_total	Total power consumption
power__leakage	Total leakage power
power__leakage__clock	Total leakage power in the clock network

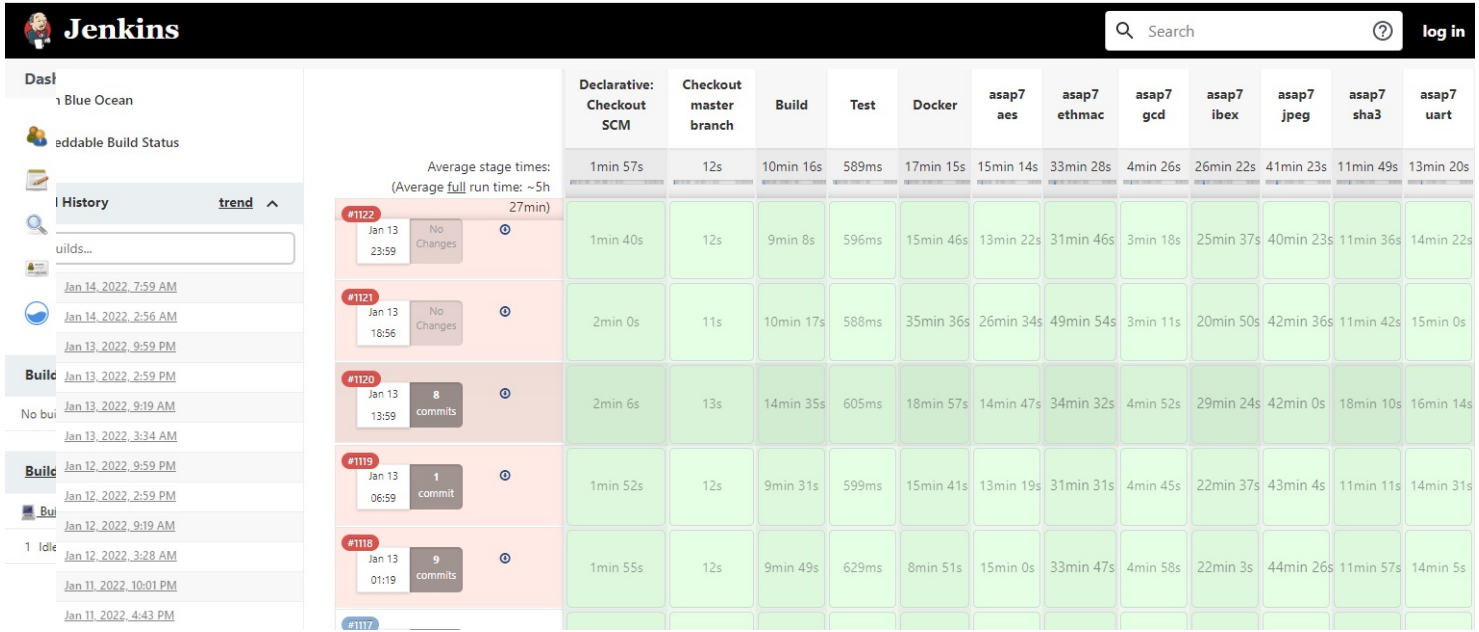
Metrics2.1 Use Model in OpenROAD

- Metrics2.1 used in OpenROAD
  - Monitor and gate the PRs into the OpenROAD repo
  - Create the daily dashboards for OpenROAD and OpenROAD flow
    - https://github.com/The-OpenROAD-Project/OpenROAD
    - https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts
  - Quantify improvements to engines and flows

- Metrics2.1 used to collect data from DoE runs
  - Metrics collection of DoE runs to support AutoTuner
  - Metrics collection of DoE runs for visualization and ML applications

OpenROAD Public Dashboards

- OpenROAD public dashboard based on Metrics2.1 is updated daily at jenkins.openroad.tools
- Multiple pipelines are run daily in cloud



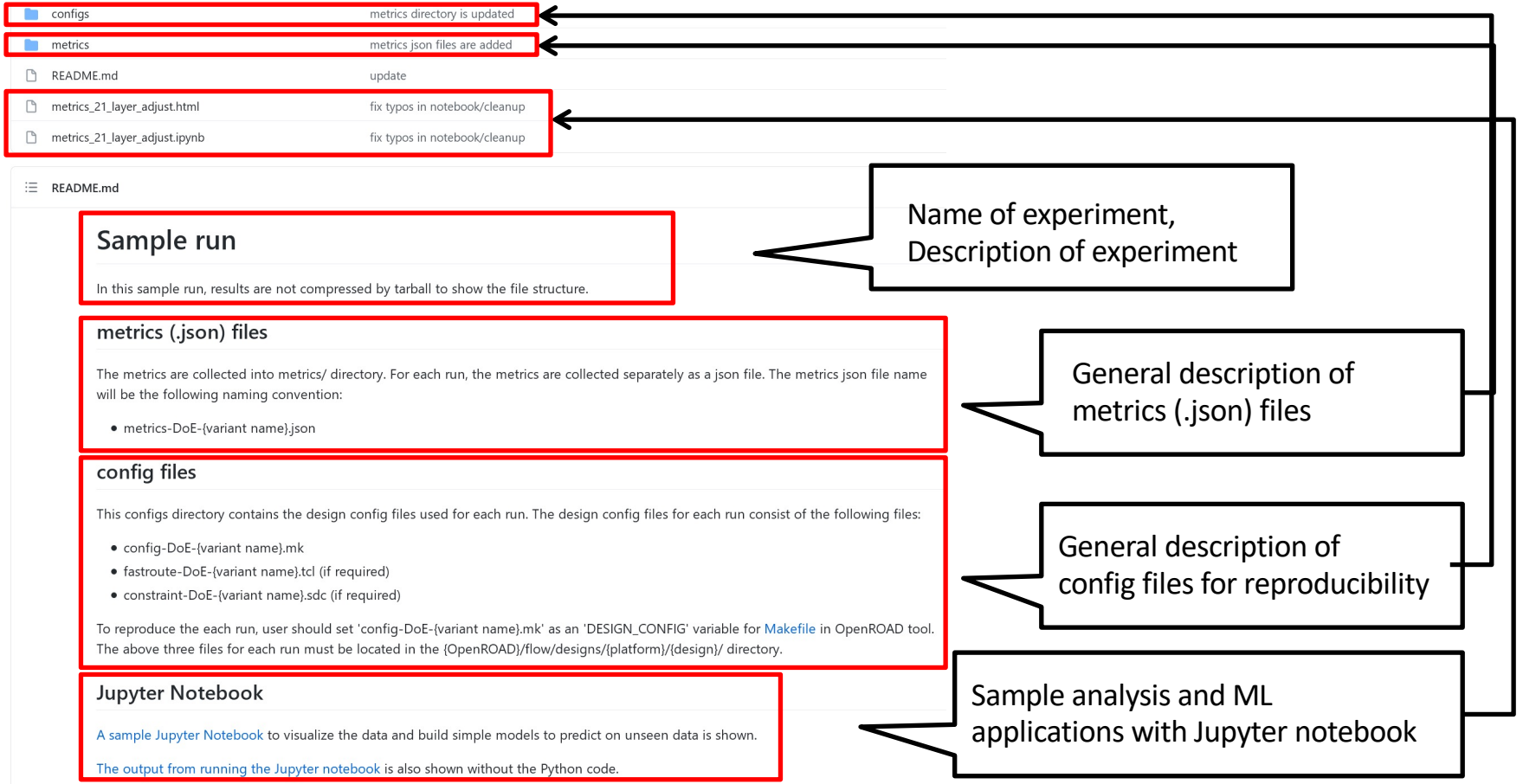
OpenROAD flow scripts “rules.json”

- OpenROAD flow scripts uses “rules.json” to determine pass or fail of designs based on different metrics.
- Currently 14 metrics are checked for each design
- rules.json can be automatically created from metrics data
- Sample rules.json for sky130HD “ibex_core” is shown


```
{
    "field": "cts__timing__setup__ws__pre_repair",
    "value": -2.75,
    "compare": ">="
},
{
    "field": "cts__timing__setup__ws__post_repair",
    "value": -2.75,
    "compare": ">="
},
{
    "field": "globalroute__timing__clock__slack",
    "value": -2.12,
    "compare": ">="
},
{
    "field": "globalroute__timing__setup__ws",
    "value": -2.12,
    "compare": ">="
},
{
    "field": "detailedroute__route__wirelength",
    "value": 915751,
    "compare": "<="
},
{
    "field": "detailedroute__route__drc_errors",
    "value": 0,
    "compare": "<="
},
{
    "field": "finish__timing__setup__ws",
    "value": -2.75,
    "compare": ">="
},
{
    "field": "finish__design__instance__area",
    "value": 231165,
    "compare": "<="
}
```

Sample Collection of DoE Runs

https://github.com/ieee-ceda-datc/datc-rdf-Metrics4ML/tree/main/experiments/sky130hd__ibex_core__sample



Sample METRICS2.1 Application: ibex, SKY130HD

- **ibex =** lowRISC RISC-V core [https://github.com/lowRISC/ibex] (15,708 instances)
- Design run with OpenROAD flow scripts
- Parameters swept:
  - Core utilization of 20% to 40% in increments of 2%
  - Layer track resource adjustment for M1 to M4 from 0.1 to 0.2 in steps of .025
- Metrics of interest:
  - Successful or Doomed (failure in Global or Detailed Route)
  - Route runtime, DRC errors
  - Routed wirelength
- Build prediction models for Success/Failure and estimated wirelength for new parameter settings

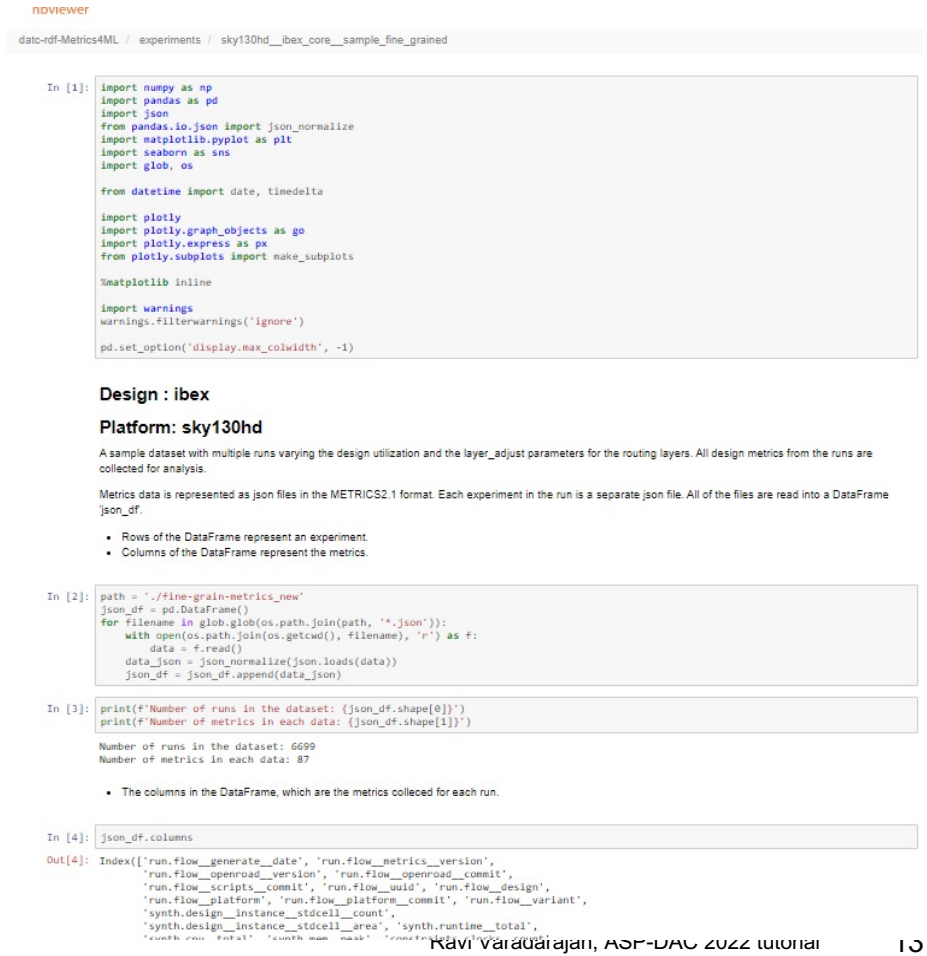
Sample METRICS2.1 Application

- https://github.com/ieee-ceda-datc/datc-rdf-Metrics4ML/tree/main/experiments/sky130hd__ibex_core__sample_fine_grained contains the data and the Jupyter Notebook

- Jupyter Notebook:
  - Read METRICS data
  - Data engineering to pull, organize relevant data
  - Visualize trends, statistics

- Leverages python packages json, pandas, matplotlib, plotly, sklearn

- Use **sklearn**
  - Logistic regression to predict successful/doomed runs
  - Linear regression to estimate wirelength, runtime



nbviewer

datc-rdf-Metrics4ML / experiments / sky130hd__ibex_core__sample_fine_grained

In [1]:
```
import numpy as np
import pandas as pd
import json
from pandas.io.json import json_normalize
import matplotlib.pyplot as plt
import seaborn as sns
import glob, os

from datetime import date, timedelta

import plotly
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots

%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_colwidth', -1)
```

Design : ibex

Platform: sky130hd

A sample dataset with multiple runs varying the design utilization and the layer_adjust parameters for the routing layers. All design metrics from the runs are collected for analysis.

Metrics data is represented as json files in the METRICS2.1 format. Each experiment in the run is a separate json file. All of the files are read into a DataFrame 'json_df'.

- Rows of the DataFrame represent an experiment.
- Columns of the DataFrame represent the metrics.

In [2]:
```
path = './fine-grain-metrics_new'
json_df = pd.DataFrame()
for filename in glob.glob(os.path.join(path, '*.json')):
    with open(os.path.join(os.getcwd(), filename), 'r') as f:
        data = f.read()
    data_json = json_normalize(json.loads(data))
    json_df = json_df.append(data_json)
```

In [3]:
```
print(f'Number of runs in the dataset: {json_df.shape[0]}')
print(f'Number of metrics in each data: {json_df.shape[1]}')
```

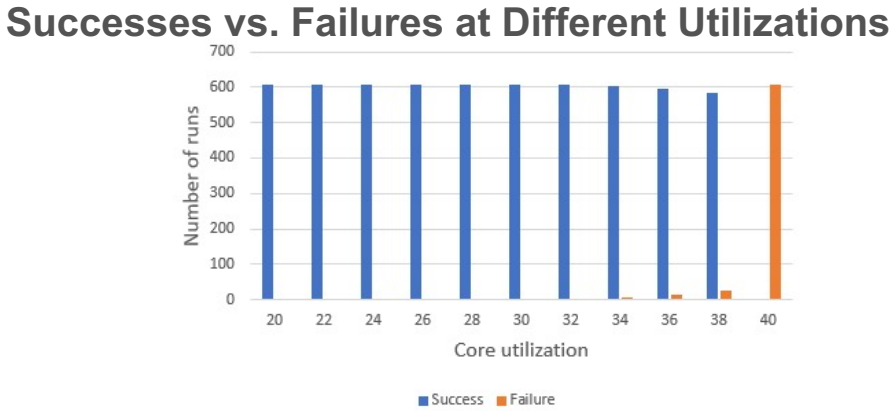
```
Number of runs in the dataset: 6699
Number of metrics in each data: 87
```

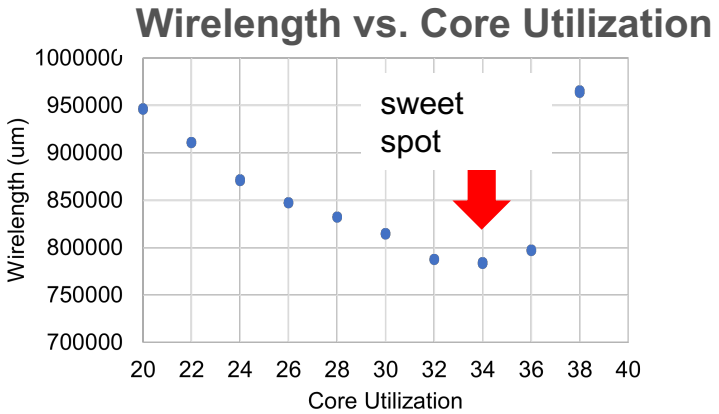
- The columns in the DataFrame, which are the metrics colleced for each run.

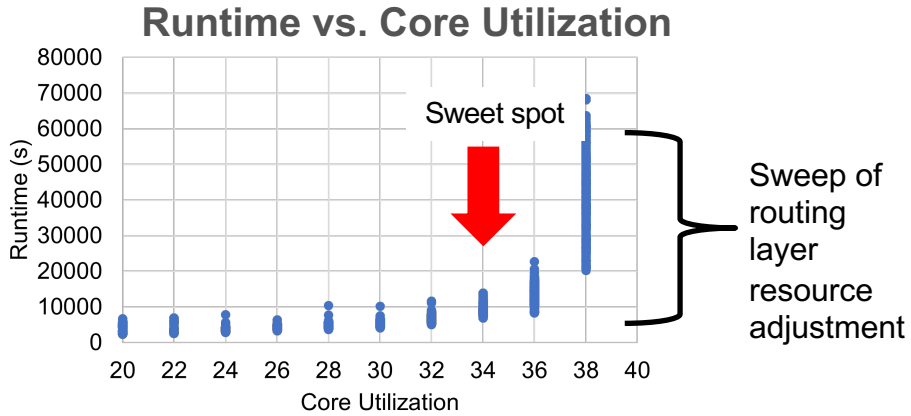
In [4]:
```
json_df.columns
```

Out[4]:
```
Index(['run.flow__generate__date', 'run.flow__metrics__version',
       'run.flow__openroad__version', 'run.flow__openroad__commit',
       'run.flow__scripts__commit', 'run.flow__uuid', 'run.flow__design',
       'run.flow__platform', 'run.flow__platform__commit', 'run.flow__variant',
       'synth.design__instance__stdcell__count',
       'synth.design__instance__stdcell__area', 'synth.runtime__total',
```

Sample Plots from Jupyter Notebook







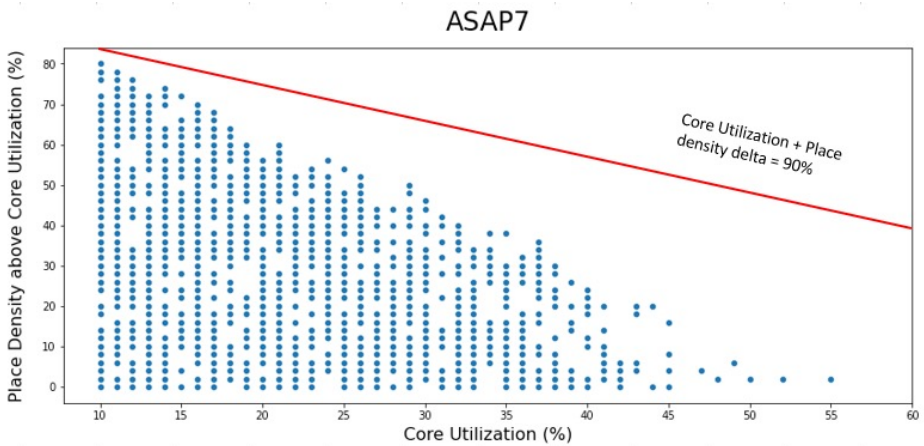
Another Sample METRICS2.1 Application

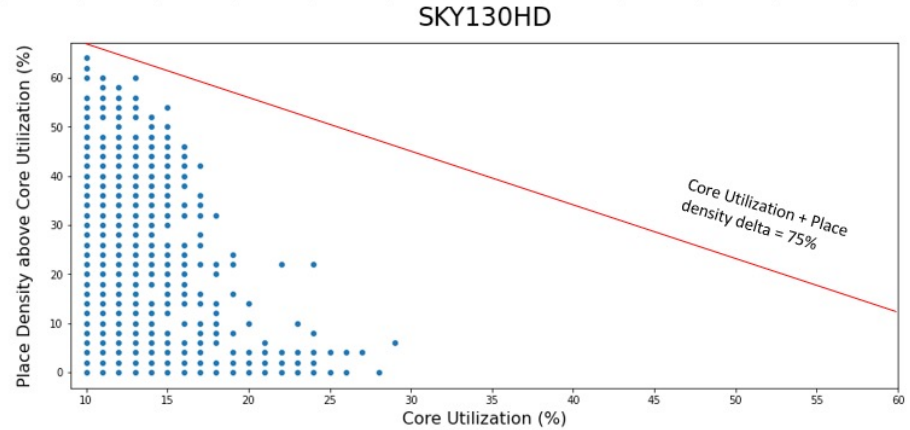
- **Design: ibex** on ASAP7 and SKY130HD platforms

- Parameters swept
  - Core Utilization: 10% to 60% in increments of 1%
  - Place Density (RePlAce) delta above Core Utilization: 0% to 90% in increments of 2%
  - 2295 data entries for each platform

- Metrics of interest
  - Successful or Doomed
  - Routed Wirelength

- Shown in Plots
  - 728 of 2295 ASAP7 runs are successful
  - 252 of 2295 SKY130HD runs are successful





Summary

- **METRICS2.1 infrastructure**
  - An initiative of the IEEE CEDA Design Automation Technical Committee (DATC)
  - Enables standardized metrics collection and design process recording
  - Used in the OpenROAD project for CI and track QoR
  - We provide archives, reproducible config files and ML applications in the form of Jupyter notebooks
- **We invite you for collaboration!**
  - METRICS2.1
    - https://github.com/ieee-ceda-datc/datc-rdf-Metrics4ML

THANK YOU ! (Questions / Follow ups?)

We thank the OpenROAD team and the IEEE CEDA Design Automation Technical Committee for many helpful interactions.

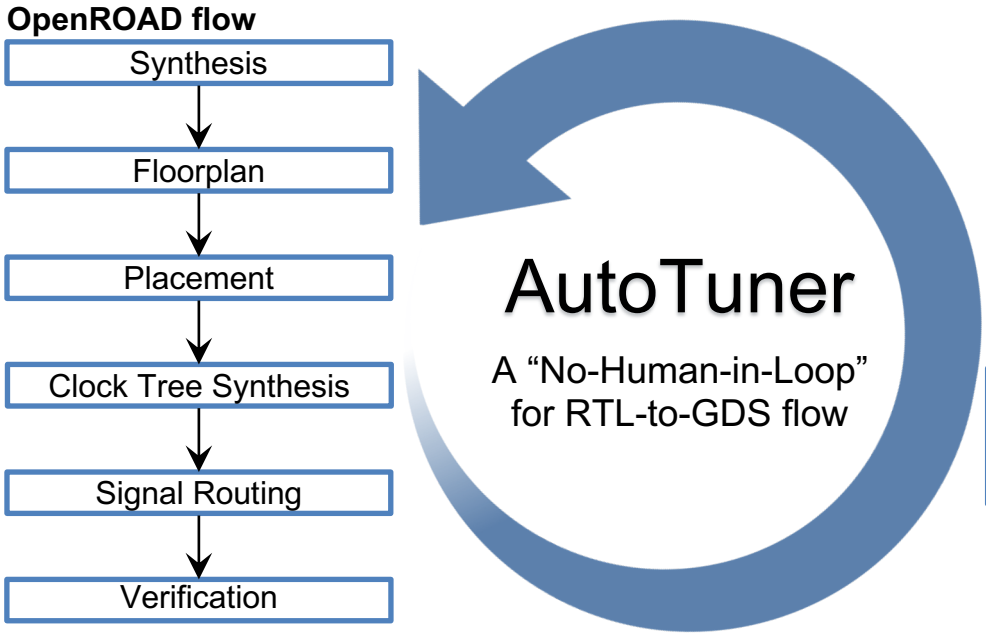
This work is supported in part by the U.S. National Science Foundation (CCF-2112665) and the U.S. Defense Advanced Research Projects Agency (HR0011-18-2-0032).
https://theopenroadproject.org https://tilos.ai https://vlsicad.ucsd.edu

Part 6

AutoTuner: Autonomous RTL-to-GDS Flow Tuning

Seungwon Kim

AutoTuner: True No-Human-in-the-Loop Operation



Best human-tuned flow
vs.
Auto-tuned flow
(No Human In Loop)

23% Faster!
45% Less Power Consumption!

Tested in SKY130HD
lowRISC RISC-V core (ibex)

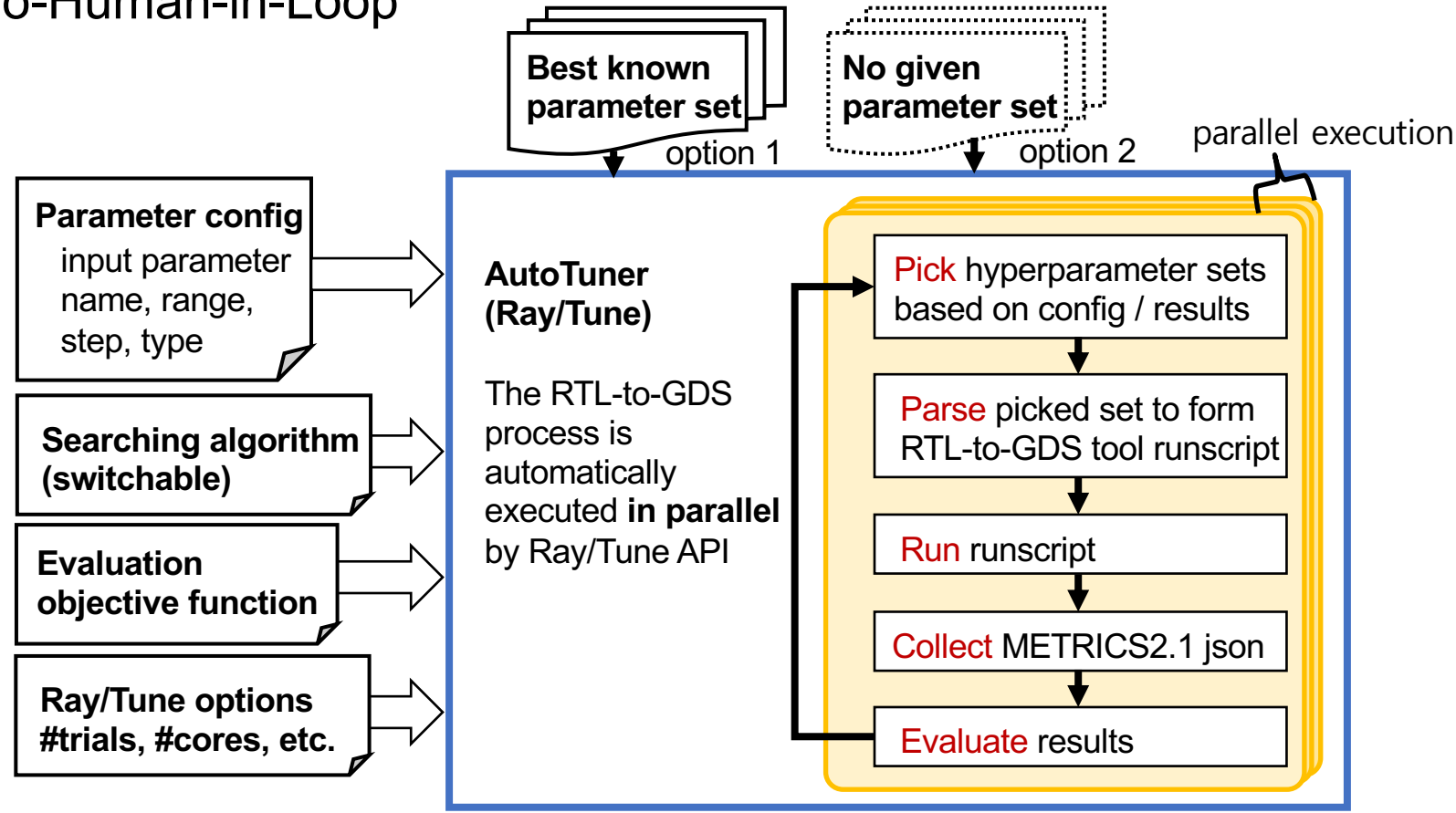
Note: AutoTuner being deployed on GCP for public enablements (SKY130*, ASAP7).

Flow Parameter AutoTuner – Overview

- A "No-Human-in-Loop" for RTL-to-GDS flow using the **OpenROAD** tool
- Automatic iterative tuning for QoR improvement within a given hyperparameter range space
- Interface: Python packages Ray/Tune
- Search algorithms: HyperOpt, PBT, Optuna, Nevergrad, Ax, random search
- Advantages:
  - Pre-existing big data is not necessary for execution
  - Fewer trials compared to random or grid parameter sweeping
  - Powerful parallelization management: core/thread management, external server usage, web visualization
  - Suboptimality gap reduces with compute resource, schedule budget

Flow Parameter AutoTuner – Architecture

- No-Human-in-Loop



Case Study: SkyWater 130HD, ibex (1/3)

- Tech: Skywater 130nm HD
- Design: ibex
- Tested Algorithms
  - HyperOpt
- Parameter config

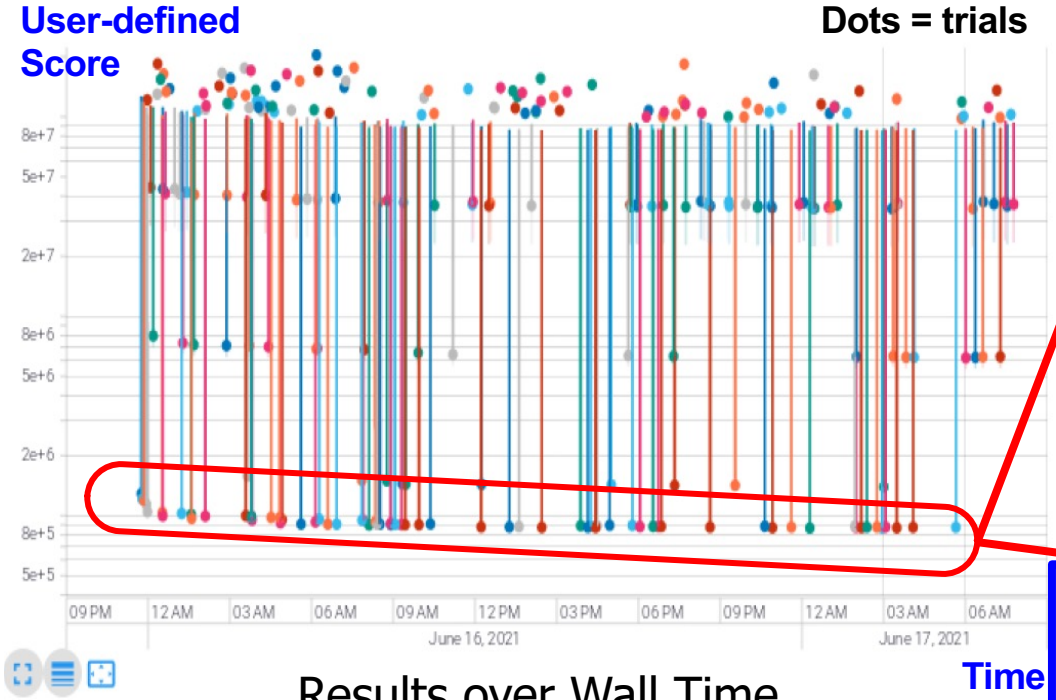
Assuming full factorial combinations,
1,058,298,150
= #possible combinations!

```
{
    "GP_PAD": {"type": "int", "minmax": [0,4], "step": 1 },
    "DP_PAD": {"type": "int", "minmax": [0,4], "step": 1 },
    "LAYER_ADJUST": {"type": "float", "minmax": [0.1,0.7], "step": 0 },
    "PLACE_DENSITY": {"type": "float", "minmax": [0.1,1.0], "step": 0 },
    "FLATTEN": {"type": "int", "minmax": [0,1], "step": 1 },
    "PINS_DISTANCE": {"type": "int", "minmax": [1,3], "step": 1 },
    "CTS_CLUSTER_SIZE": {"type": "int", "minmax": [10,40], "step": 1 },
    "CTS_CLUSTER_DIAMETER": {"type": "int", "minmax": [80,120], "step": 1 },
    "GR_OVERFLOW": {"type": "int", "minmax": [1,1], "step": 0 }
}
```

- HyperParameter name, type, minmax, step.
- When type is int and step = 0, it means constant value
- When type is float and step = 0, it means continuous range

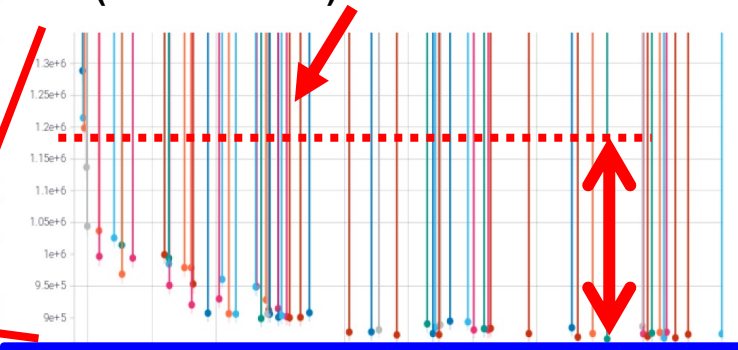
Case Study: SkyWater 130HD, ibex (2/3)

- GUI feature is supported with **TensorBoard**
- Results over Wall Time



Results over Wall Time

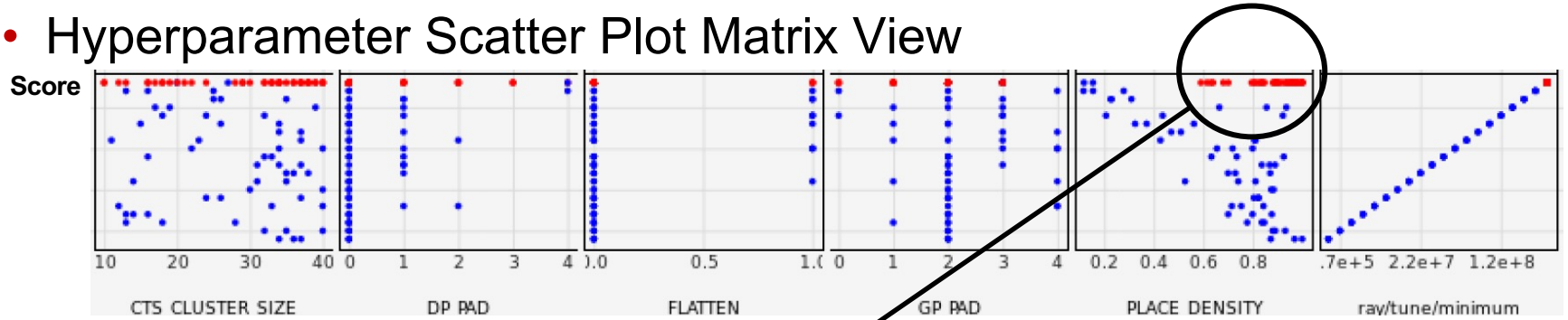
Default flow score = 1,174,346
Our Best Score = 855,373
(370 trials in total 500 #trials)
(less is better)



Improvement
WL 1003801um → 843258um **(-16%)**
Effective CP 20.935ns → 16.185 ns **(-23%)**
Total power 0.024 W→ 0.0133 W **(-45%)**

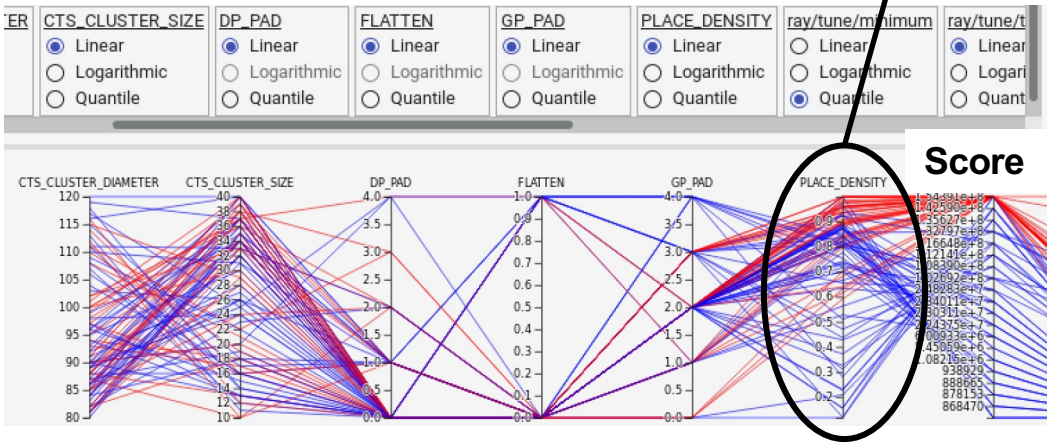
Case Study: SkyWater 130HD, ibex (3/3)

- Hyperparameter Scatter Plot Matrix View



- Parallel Coordinates View

high risk, high return



Parameters	Human	AutoTuner
Flatten (0 = hier)	1	0
Global placement padding	4	2
Detailed placement padding	2	0
IO pin distance (um)	2	2
CTS clustering size (um)	30	37
CTS clustering diameter (um)	100	95
Layer resource adjustment	0.5	0.42
Target GP place density	**0.6**	**0.99**

Search Algorithms

- **Types of search algorithms currently supported by AutoTuner**

Types	Name	Year	Author	Affiliation
Random/Grid search	Random /Grid search			
Population Based Training	PBT	2017	M. Jaderberg	Google Deepmind
Tree Parzen Estimator (TPE)	HyperOpt	2013	J. Bergstra	Rowland Institute at Harvard
Bayesian + Multi-Armed Bandit	AxSearch	2019	B. Letham	Facebook
TPE + CMA-ES†	Optuna	2019	T. Akiba	Preferred Network, Inc.
Evolutionary Algorithm (Gradient-free optimization)	Nevergrad	2018	J. Rapin	Facebook

†CMA-ES: Covariance matrix adaptation evolution strategy

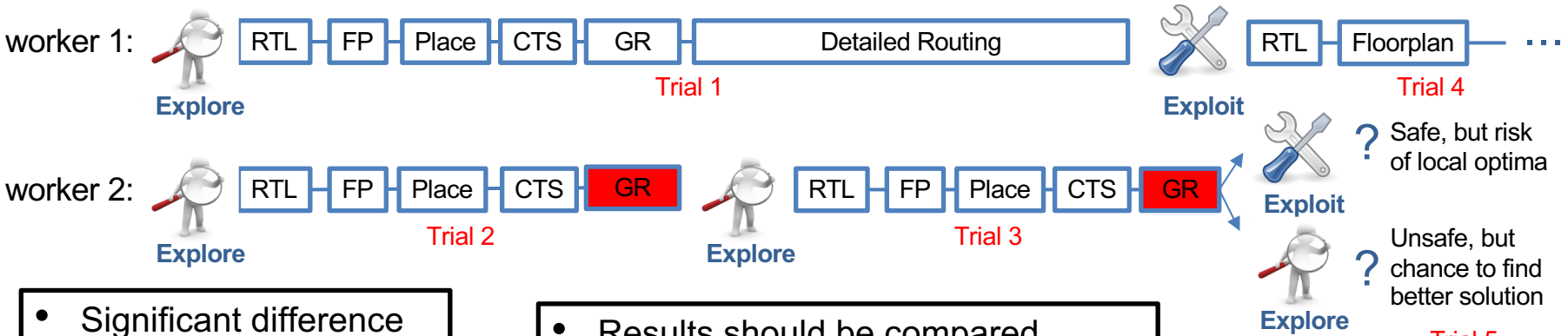
- **Each algorithm has its pros and cons depending on:**
  - Difficulty of the target black box problem
  - Number of hyperparameters
  - Type of hyperparameters (continuous, discrete, mixed)
  - Parallelization, and whether synchronous / asynchronous

Search Algorithm Usage Strategy: Considerations (1)

- Considerations for **RTL-to-GDS** domain
  - **Problem size**: Big
    - Long runtime for each trial
  - **Difficult**: Many infeasible input parameter combinations for hyperparameters
    - Active exploration is required to find feasible starting points
  - **Mixed types**: continuous and discrete parameters
    - Some Bayesian optimization algorithms do not support discrete parameters
  - **CAD Tool noise**: influence on the final QoR from very small variations in input values

Search Algorithm Usage Strategy: Considerations (2)

- Considerations for algorithm selection
  - Most time-consuming stage is detailed routing
  - The runtime required per trial varies greatly
  - Each algorithm has different strategy of exploration and exploitation



- Significant difference in total walltime with **same #trials**
- Many infeasible hyperparameter sets

- Results should be compared by the **same walltime constraint**
- Active exploration with **asynchronous scheduling** helps

Objective Function – Power, Performance and Area (PPA)

- IC design is a multi-objective optimization (e.g., power, performance and area)
- We integrate three PPA metrics into one objective function
  - To increase the general availability
  - To use algorithms that do not support multi-objective functions
- User-settable coefficient values ($C_{power} \cdot C_{perform} \cdot C_{area}$) of three objectives to set the direction of tuning

Minimize $\quad Score = (PPA_{imp}^{UB} - PPA_{imp}) \cdot (1 + \alpha \cdot N_{DRV})$ $\quad$ α: penalty factor (default: 0.1)

s.t.

$$PPA_{imp} = (C_{power} \cdot Power_{imp} + C_{perform} \cdot Perform_{imp} + C_{area} \cdot Area_{imp})$$

$$PPA_{imp}^{UB} = (C_{power} + C_{perform} + C_{area})$$

$$Power_{imp} = (P_{ref} - P)/P_{ref}$$

$$Perform_{imp} = (effCP_{ref} - effCP)/effCP_{ref}$$

$$Area_{imp} = (A_{ref} - A)\}/(A_{ref})$$

P : Total power
effCP: Effective clock period
A: Cell area
ref: Reference metric

Demonstration of AutoTuner

- AutoTuner provides two main functionalities as follows
  - **Automatic hyperparameter tuning framework** for OpenROAD-flow-scripts
  - **Parametric sweeping experiments** for OpenROAD-flow-scripts
- Installation of **OpenROAD-flow-scripts (ORFS)** is required
  - https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts
- In this demonstration, the following processes are included
  - Tunable / sweepable parameters
  - How to define inputs
    - Input JSON structure (autotuner.json)
    - User-settable coefficient values (distributed.py)
  - List of input arguments
  - How to run in shell by AutoTuner in ORFS
  - How to find the logs and results
  - How to use GUI
- Instructions for AutoTuner
  - https://github.com/The-OpenROAD-Project/OpenROAD/blob/master/docs/user/InstructionsForAutoTuner.md

Demonstration – Tunable / Sweepable Parameters (1)

- **Any variable** that can be set from the command line can be used
  - Technology platform config variables (with "?=" type definition)
  - Design config variables
  - The other environmental variables used in ORFS
- **Technology platform variables** (e.g., ASAP 7nm)
  - Location of technology platform variables

config file: OpenROAD-flow-scripts/flow/platforms/asap7/config.mk

  - Example variables defined by type "?="

```
# Cell padding in SITE widths to ease rout-ability.  Applied to both sides
export CELL_PAD_IN_SITES_GLOBAL_PLACEMENT ?= 2
export CELL_PAD_IN_SITES_DETAIL_PLACEMENT ?= 1
```

- The reason why we need to define "?=" type for platform config
  - ORFS config reading order: design config → AutoTuner→ platform config
  - "?=" only has an effect if the variable is not yet defined

Demonstration – Tunable / Sweepable Parameters (2)

- **Design config variables** (e.g., aes design, ASAP 7nm)
  - Location of design config variables

config file: OpenROAD-flow-scripts/flow/designs/asap7/aes/config.mk

  - Example variables

```
export CORE_UTILIZATION       = 30
export CORE_ASPECT_RATIO      = 1
export CORE_MARGIN            = 2
```

- Duplicated design config variables will be overwritten by AutoTuner

Demonstration – Tunable / Sweepable Parameters (3)

- **The other environmental variables (1)**
  - Location of flow scripts in ORFS

`scripts path: OpenROAD-flow-scripts/flow/scripts`

  - Example variable in flow script (detailed_route.tcl):

```
if { [info exists ::env(OR_SEED)]} {
  append additional_args " -or_seed $::env(OR_SEED)"
```

  - As above example, any env variable in flow scripts is Tunable / Sweepable

Demonstration – Tunable / Sweepable Parameters (4)

- **The other environmental variables (2)**
  - For SDC:

_SDC_FILE_PATH	Path relative to the current JSON file to the SDC file.
_SDC_CLK_PERIOD	Design clock period. This will create a copy of **_SDC_FILE_PATH** and modify the clock period.
_SDC_UNCERTAINTY	Clock uncertainty. This will create a copy of **_SDC_FILE_PATH** and modify the clock uncertainty.
_SDC_IO_DELAY	I/O delay. This will create a copy of **_SDC_FILE_PATH** and modify the I/O delay.

  - For FastRoute (global router):

_FR_FILE_PATH	Path relative to the current JSON file to the **fastroute.tcl** file.
_FR_LAYER_ADJUST	Layer adjustment. This will create a copy of **_FR_FILE_PATH** and modify the layer adjustment for all routable layers, i.e., from **$MIN_ROUTING_LAYER** to **$MAX_ROUTING_LAYER**.
_FR_LAYER_ADJUST_NAME	Layer adjustment for layer NAME. This will create a copy of **_FR_FILE_PATH** and modify the layer adjustment only for the layer NAME.
_FR_GR_SEED	Global route random seed. This will create a copy of **_FR_FILE_PATH** and modify the global route random seed.

Demonstration – How to define inputs (1)

- **Input JSON structure (autotuner.json)**
  - A part of sample JSON file for ASAP 7nm aes design:

```
{
  "_SDC_FILE_PATH"  "constraint.sdc",
  "_S                ERIOD"
        "type"
        "minmax"  [
                  100,
                  600
        ]
        "step"  0
  }
  "CORE_MARGIN"  {
        "type"  "int",
        "minmax"  [
                  2,
                  2
        ]
        "step"  0
  },
  ...
}
```

- **Parameter names for sweeping/tuning**
- **Parameter type ("float" or "int") for sweeping/tuning**
- **Min-to-max range for sweeping/tuning. The unit follows the default value of each technology std cell library**
- **Parameter step within the minmax range.**
  Step 0 for type "float" means continuous step.
  Step 0 for type "int" means a constant parameter

path: OpenROAD-flow-scripts/flow/designs/asap7/aes/autotuner.json

Demonstration – How to define inputs (2)

- **User-settable coefficient values (distributed.py)**
  - A part of distributed.py :

```
class PPAImprov(AutoTunerBase):
    '''
    PPAImprov
    '''

    @classmethod
    def get_ppa(cls, metrics):
        '''
        Compute PPA term for evaluate.
        '''
        coeff_perform, coeff_power, coeff_area = 10000, 100, 100
```

  - Function “get_ppa” in class “PPAImprove” contains coefficient variables
  - User can adjust three coefficient to set the PPA optimization direction of tuning

path: OpenROAD-flow-scripts/flow/util/distributed.py

Demonstration – List of input arguments (1)

- **Input arguments are classified into five categories**
  - Target Design
  - Experiment Setup
  - Git Setup
  - AutoTuner Specific Arguments
  - Workload
- For arguments with default settings, refer to distributed.py

Demonstration – List of input arguments (2)

- **Target Design**
  - --design
    - Name of the design for autotuning
  - --platform
    - Name of the platform for autotuning

- **Experiment setup**
  - --config
    - Configuration file that sets which knobs to use for autotuning
  - --experiment
    - Experiment name. This parameter is used to prefix the **FLOW_VARIANT** and to set the Ray log destination

Demonstration – List of input arguments (3)

- **Git setup**
  - --git-clean
    - Clean binaries and build files
  - --git-clone
    - Force new git clone
  - --git-clone-args
    - Additional git clone arguments
  - --git-latest
    - Use the latest version of OR app
  - --git-or-branch
    - OR app branch to use
  - --git-orfs-branch
    - ORFS branch to use
  - --git-url
    - ORFS repo URL to use
  - --build-args
    - Additional arguments given to **./build_openroad.sh**

Demonstration – List of input arguments (4)

- **AutoTuner Specific Arguments**
  - --algorithm
    - Search algorithm to use for autotuning
  - --eval
    - Evaluate function to use with search algorithm
  - --samples
    - Number of samples for autotuning
  - --iterations
    - Number of iterations for autotuning
  - --resume
    - Resume previous run
  - --reference
    - Reference METRICS2.1 json for "PPAImprov" evaluation
    - **If not defined**, "AutoTunerBase“ will be used (performance opt only)
  - --perturbation
    - Perturbation interval for PopulationBasedTraining
  - --seed
    - Random seed for parameter selection during autotuning

Demonstration – List of input arguments (5)

- **Workload**
  - --jobs
    - Max number of concurrent jobs
  - --openroad-threads
    - Max number of threads OR app can use
  - --server
    - The address of Ray server to connect
  - --port
    - The port of Ray server to connect
- -v, --verbose
  - Verbosity level
    - 0: only print Ray status
    - 1: also print training stderr
    - 2: also print training stdout

Demonstration – How to run in the shell (1)

- **Run in Shell**
  - **distributed.py** scripts handles sweeping and tuning of ORFS parameters

  - AutoTuner run example (SKY130nm HD ibex design):

```
shell) python3 distributed.py --design ibex --platform sky130hd \
       --config ../designs/sky130hd/ibex/autotuner.json \
       --jobs 40 --openroad-threads 4 \
      tune --samples 1000
```

  - The order of the parameters matter
    - Arguments **--design**, **--platform** and **--config** are always required and should precede
    - **AutoTuner specific arguments** must be typed after **"tune"**

```
launch path: OpenROAD-flow-scripts/flow/util
```

Demonstration – How to run in the shell (2)

```
shell) python3 distributed.py --design ibex --platform sky130hd \
       --config ../designs/sky130hd/ibex/autotuner.json \
       --jobs 40 --openroad-threads 4 \
      tune --samples 1000
```

Launch

```
(swkim38) [swkim@autotuner-1 util]$ python3 distributed.py --design ibex --platform sky130hd      --config ../designs/sky130hd/ibex/autotuner.j
son       --jobs 40 --openroad-threads 4  tune --samples 1000
2022-01-04 13:44:22,736 INFO services.py:1272 -- View the Ray dashboard at http://127.0.0.1:8265
== Status ==
Memory usage on this node: 120.9/3783.4 GiB
Using AsyncHyperBand: num_stopped=0
Bracket: Iter 64.000: None | Iter 16.000: None | Iter 4.000: None | Iter 1.000: 
Resources requested: 0/160 CPUs, 0/0 GPUs, 0.0/3478.44 GiB heap, 0.0/186.26 GiB objects
Result logdir: /home/swkim/01_OpenROAD/daily_runs/OpenROAD_daily_2021-12-28_00-07/OpenROAD-flow-scripts/flow/logs/sky130hd/ibex/test-tune-2022-01
-04-13-44-21
Number of trials: 1/1000 (1 PENDING)
| Trial name              | status    | loc    |   CELL_PAD_IN_SITES_DETAIL_PLACEMENT |   CELL_PAD_IN_SITES_GLOBAL_PLACEMENT |   CORE_ASPECT_RATIO |
   CORE_MARGIN |    CORE_UTILIZATION |   CTS_CLUSTER_DIAMETER |   CTS_CLUSTER_SIZE |   PLACE_DENSITY_LB_ADDON |   _FR_GR_OVERFLOW |   _FR_LAYER_AD
JUST |    _PINS_DISTANCE |    _SDC_CLK_PERIOD |    _SYNTH_FLATTEN |
| AutoTunerBase_6e9e8a08 | PENDING  |        |                                    1 |                                    4 |            0.860483 |
             2 |                  98 |                    116 |                 20 |                 0.763522 |                 1 |           0.62
4004 |                 2 |             10.1019 |                  1 |
```

status of server and tunning

status of initial trial

Demonstration – How to run in the shell (2)

```
shell) python3 distributed.py --design ibex --platform sky130hd \
       --config ../designs/sky130hd/ibe
       --jobs 40 --openroad-threads 4
```

In progress of autotuning

```
Current best trial: 6e9e8a08 with minimum=9999999999900.0 and parameters={'_SDC_CLK_PERIOD': 10.101861658418082, 'CORE_UTILIZATION': 98, 'CORE_AS
PECT_RATIO': 0.8604828491607834, 'CORE_MARGIN': 2, 'CELL_PAD_IN_SITES_GLOBAL_PLACEMENT': 4, 'CELL_PAD_IN_SITES_DETAIL_PLACEMENT': 1, '_FR_LAYER_A
DJUST': 0.6240038814181154, 'PLACE_DENSITY_LB_ADDON': 0.7635223697775535, '_SYNTH_FLATTEN': 1, '_PINS_DISTANCE': 2, 'CTS_CLUSTER_SIZE': 20, 'CTS_
CLUSTER_DIAMETER': 116, '_FR_GR_OVERFLOW': 1}
Result logdir: /home/swkim/01_OpenROAD/daily_runs/OpenROAD-flow-scripts/flow/logs/sky130hd/ibex/autotune-2022-01
-04-13-44-21
Number of trials: 47/1000 (39 RUNNING, 8 TERMINATED)
```

current progress

current best

status of trials

Trial name	status	loc	CELL_PAD_IN_SITES_DETAIL_PLACEMENT	CELL_PAD_IN_SITES_GLOBAL_PLACEMENT	CORE_ASPECT_RATIO	CORE_MARGIN	CORE_UTILIZATION	CTS_CLUSTER_DIAMETER	CTS_CLUSTER_SIZE	PLACE_DENSITY_LB_ADDON	_FR_GR_OVERFLOW	_FR_LAYER_ADJUST	_PINS_DISTANCE	_SDC_CLK_PERIOD	_SYNTH_FLATTEN	iter	total time (s)	minimum
AutoTunerBase_6ec6d760	RUNNING		4		0.571708	2	57	83	36	0.9								0.398468
						1	1.94542	0										
AutoTunerBase_6eda5bfa	RUNNING		2	3	1.55892	2	53	112	20	0.825331	1	0.381646						
						2	3.86515	1										
AutoTunerBase_6f140e04	RUNNING		3	2	1.45467	2	37	85	26	0.879843	1	0.636041						
						2	4.87573	1										
AutoTunerBase_6f277958	RUNNING		2	1	0.486949	2	76	112	15	0.0445132	1	0.567149						
						1	9.58792	1										
AutoTunerBase_6f619dcc	RUNNING		3	3	0.866958	2	20	108	24	0.869995	1	0.31722						
						2	5.49068	1										

Demonstration – How to run in the shell (3)

```
shell) python3 distributed.py --design ibex --platform sky130hd \
       --config ../designs/sky130hd/ibex/autotuner.json \
       --jobs 40 --openroad-threads 4 \
      tune --samples 1000
```

Finish

```
| AutoTunerBase_fb1ebc82 | TERMINATED |        |                    0 |                  3 |        1.64403
 |          2 |              23 |                116 |              30 |      0.472811    |             1 |          0.
170681 |          3 |         8.34024 |          0 |       1 |      4177.21     |  1515.02   |
| AutoTunerBase_931776e0 | TERMINATED |        |                    0 |                  2 |        1.71203
 |          2 |              28 |                118 |              29 |      0.521689    |             1 |          0.
278662 |          3 |         9.18671 |          0 |       1 |     17583.3      |    1e+13 |
| AutoTunerBase_d8ff999e | TERMINATED |        |                    0 |                  3 |        1.50782
 |          2 |              25 |                114 |              31 |      0.523669    |             1 |          0.
120043 |          3 |         8.03638 |          0 |       1 |      8425.05     |  1556.64   |
| AutoTunerBase_bee64372 | TERMINATED |        |                    0 |                  2 |        1.60813
 |          2 |              21 |                111 |              28 |      0.464447    |             1 |          0.
20329  |          3 |         8.66951 |          0 |       1 |      3336.07     |  1728.95   |
+----------------------+------------+--------+-------------------------------------+--------------------+-----------
-+--------------+-------------------+------------------------+--------------------+------------
-------+-----------------+-----------------+-----------------+--------+-----------------+---

2022-01-05 13:58:10,963 INFO tune.py:549 -- Total run time: 87225.26 seconds (87224.14 seconds for the tuning loop).
[INFO TUN-0002] Best parameters found: {'_SDC_CLK_PERIOD': 8.64722781707361, 'CORE_UTILIZATION': 20, 'CORE_ASPECT_RATIO': 1.5685476008329589, 'CO
RE_MARGIN': 2, 'CELL_PAD_IN_SITES_GLOBAL_PLACEMENT': 3, 'CELL_PAD_IN_SITES_DETAIL_PLACEMENT': 0, '_FR_LAYER_ADJUST': 0.10066503378702243, 'PLACE_
DENSITY_LB_ADDON': 0.46158507922619585, '_SYNTH_FLATTEN': 0, '_PINS_DISTANCE': 3, 'CTS_CLUSTER_SIZE': 31, 'CTS_CLUSTER_DIAMETER': 117, '_FR_GR_OV
ERFLOW': 1}
(pid=162978) [INFO TUN-0003] Best parameters written to logs/sky130hd/ibex/test-tune-2022-01-04-13-44-21/autotuner-best.json
(swkim38) [swkim@autotuner-1 util]$
```

best parameters

Demonstration – How to find the logs and results

- **Logs**

```
logs path: OpenROAD-flow-
scripts/flow/logs/sky130hd/ibex/test-tune-2022-01-04-12-09-50
```

Platform Design Exp name Mode Date

AutoTuner logs for each trial

Current state of tuning for checkpoint and logging

OR logs for each trial (**hash** can be found in **variant_hash.txt** in AutoTuner logs)

Demonstration – How to find the logs and results

- **Results**

```
results path: OpenROAD-flow-scripts/flow/results/sky130hd/ibex/test-tune-2022-01-04-12-09-50
{results path}/variant-3603bf82cd6b0e4099334169055 0d8af
```

hash

```
[swkim@autotuner-1 variant-3603bf82cd6b0e409933416905 50d8af]$ ls
1_1_yosys.v                2_floorplan.def        4_1_cts.def
1_synth.sdc                2_floorplan.sdc        4_2_cts_fillcell.def
1_synth.v                  2_floorplan.v          4_cts.def
2_1_floorplan.def          3_1_place_gp.def       4_cts.sdc
2_2_floorplan_io.def       3_2_place_iop.def      4_cts.v
2_3_floorplan_tdms.def     3_3_place_resized.def  output_guide.mod
2_4_floorplan_macro.def    3_4_place_dp.def       route.guide
2_5_floorplan_tapcell.def  3_place.def            updated_clks.sdc
2_6_floorplan_pdn.def      3_place.sdc
```

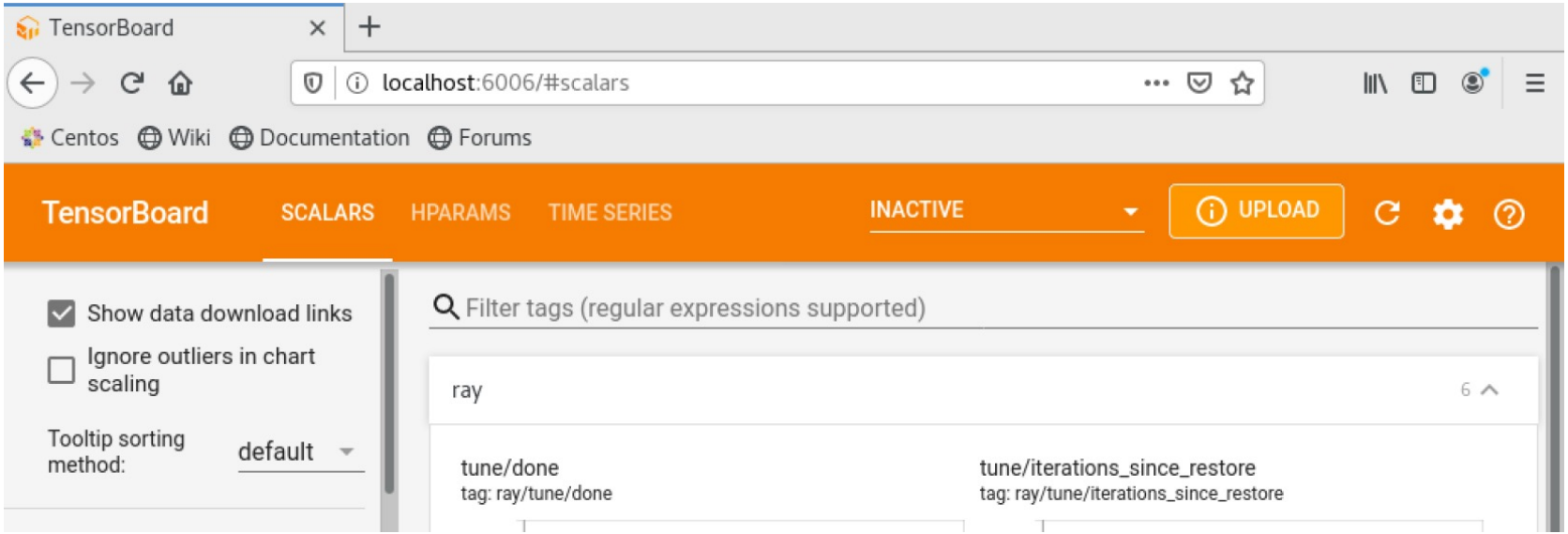
Intermediate/final DEFs, SDCs, GR guides, and Verilog files

Demonstration – How to use GUI (Tensorflow GUI) (1)

- **How to run Tensorflow GUI**
  - **Run Tensorboard**

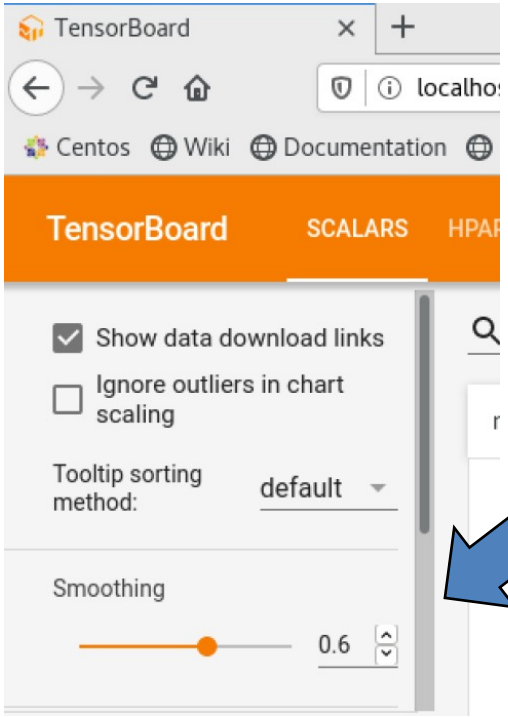
```
shell) cd {log paths}
shell) tensorboard --logdir=./
```

  - **Open the local webpage (http://localhost:6006)**

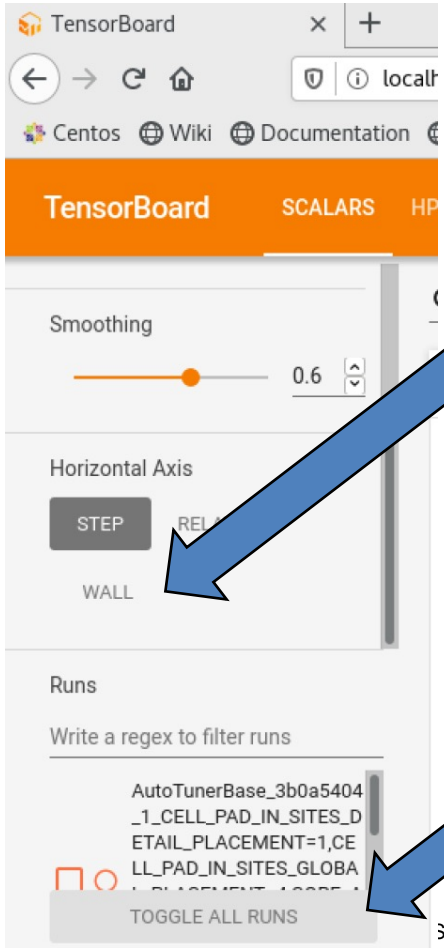


Demonstration – How to use GUI (Tensorflow GUI) (2)

- **How to see the results**
  - **Results over walltime**



(1) scroll down

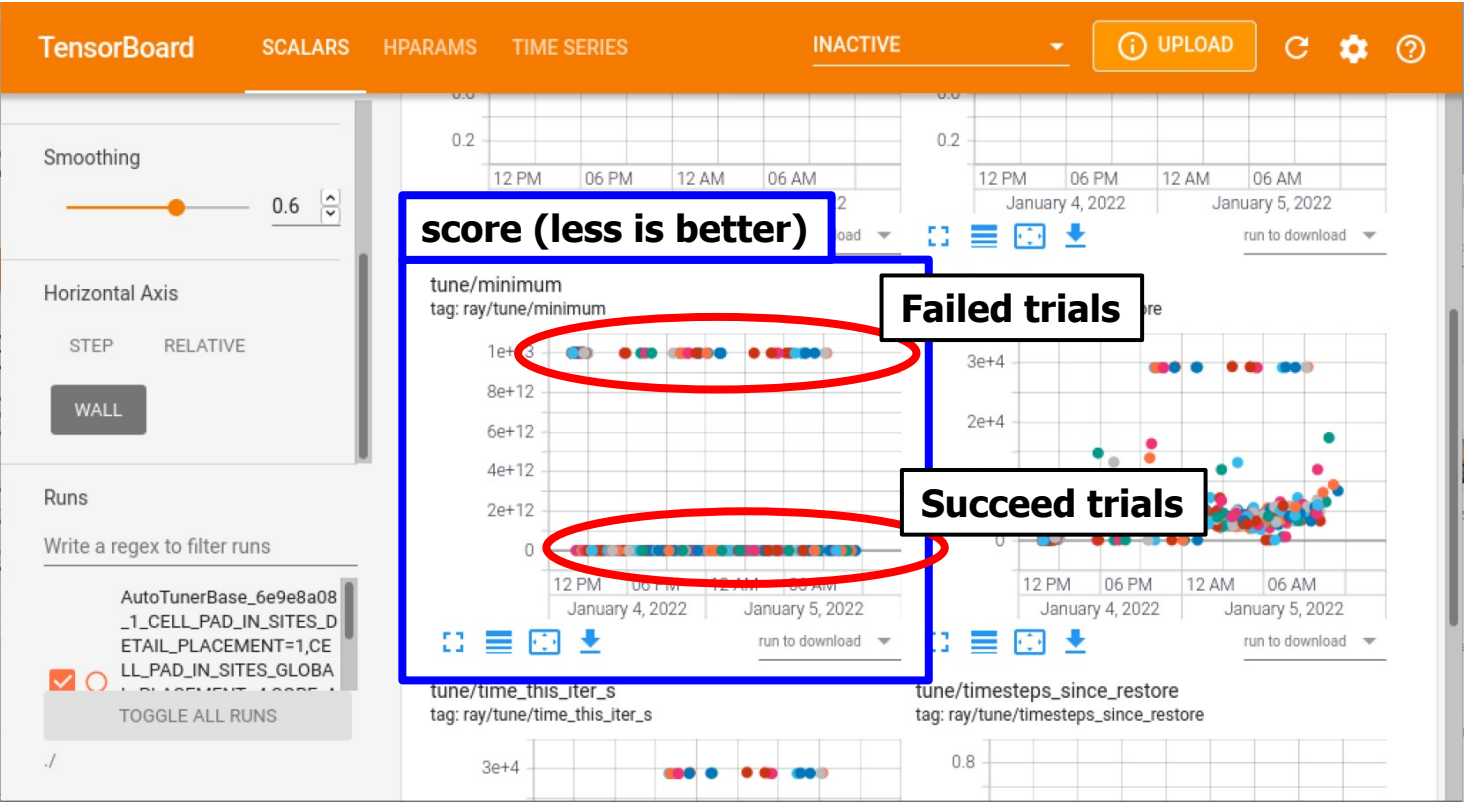


(2) click WALL

(3) click TOGGLE ALL RUNS

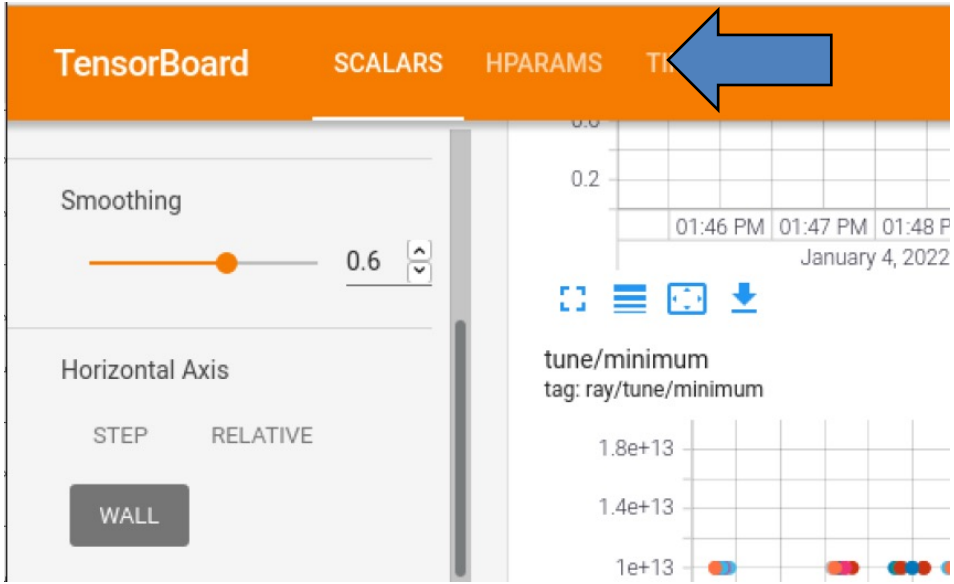
Demonstration – How to use GUI (Tensorflow GUI) (3)

- How to see the results
  - Results over walltime



Demonstration – How to use GUI (Tensorflow GUI) (4)

- **How to see the results**
  - **Parallel Coordinates View**

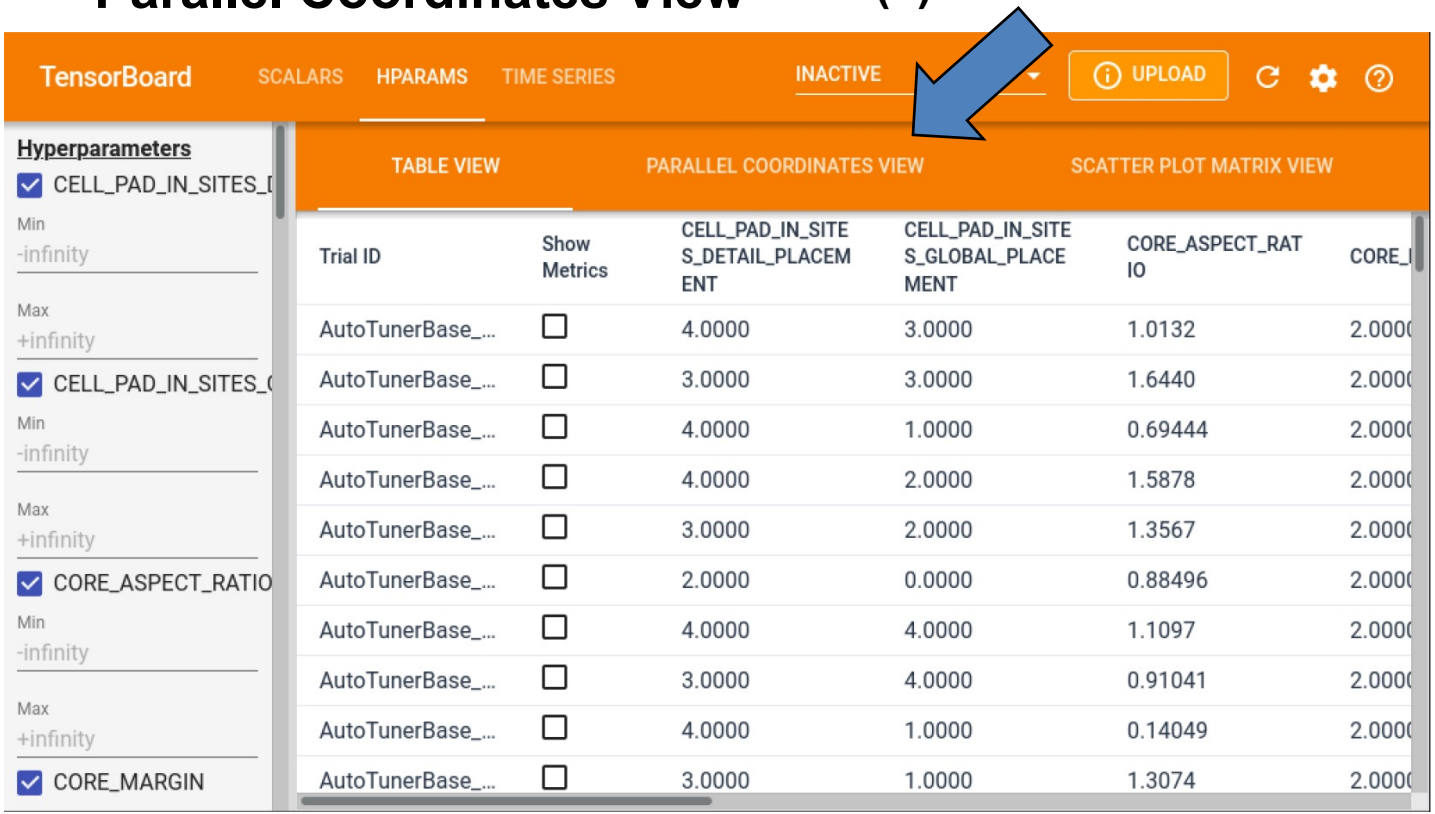


(1) click HPARAMS

Demonstration – How to use GUI (Tensorflow GUI) (5)

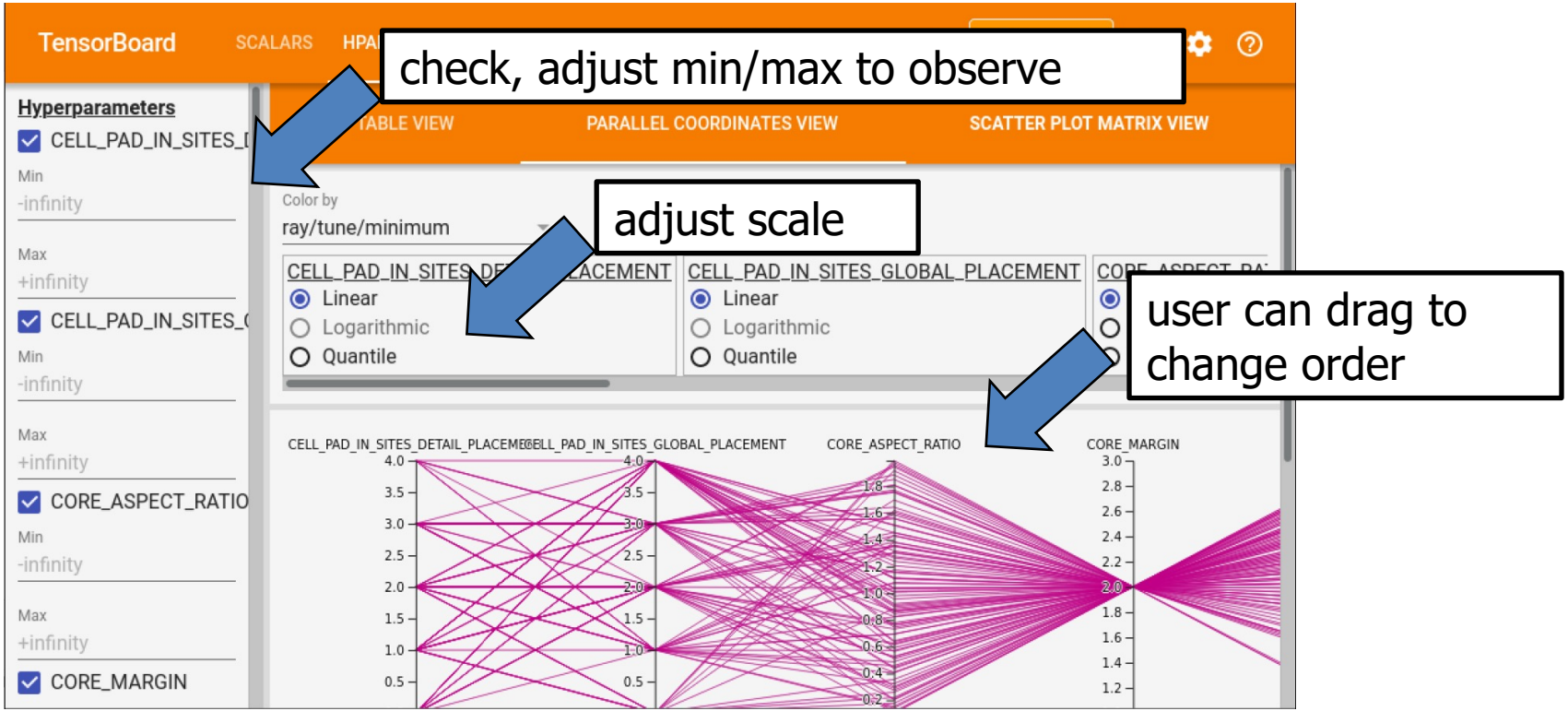
- **How to see the results**
  - **Parallel Coordinates View**

(2) click PARALLEL COORDINATES VIEW



Demonstration – How to use GUI (Tensorflow GUI) (6)

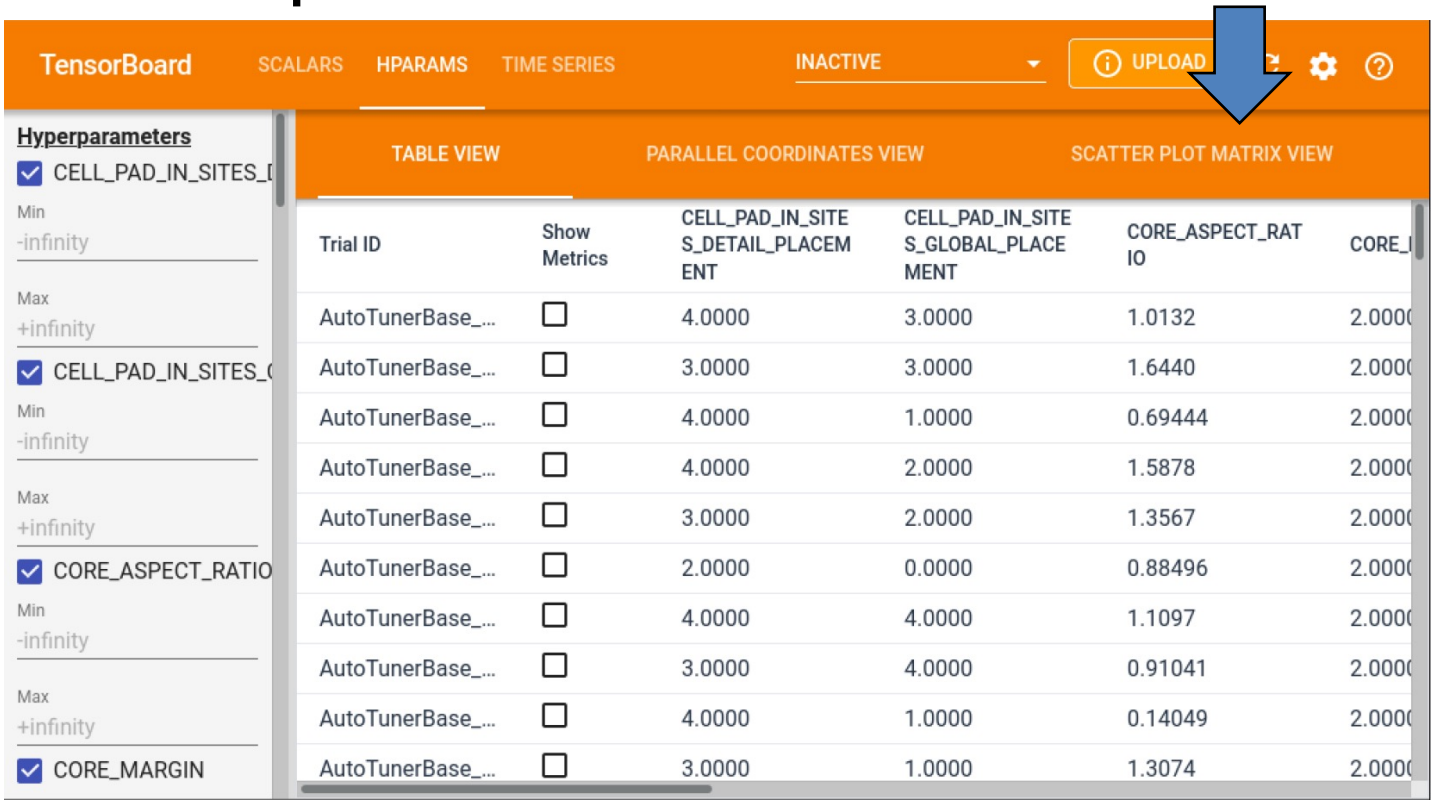
- **How to see the results**
  - **Parallel Coordinates View**



Demonstration – How to use GUI (Tensorflow GUI) (7)

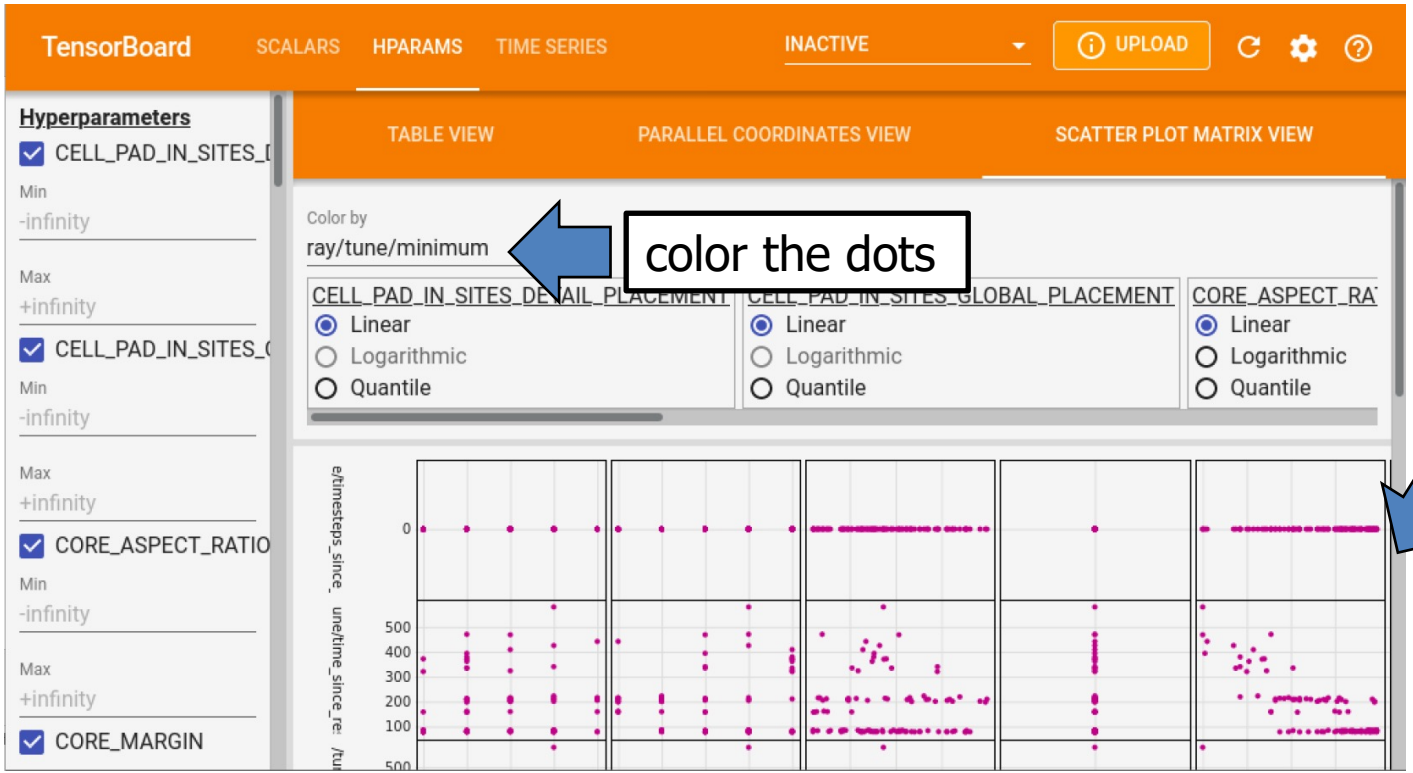
- **How to see the results**
  - **Scatter plot matrix**

(1) click SCATTER PLOT MATRIX VIEW



Demonstration – How to use GUI (Tensorflow GUI) (8)

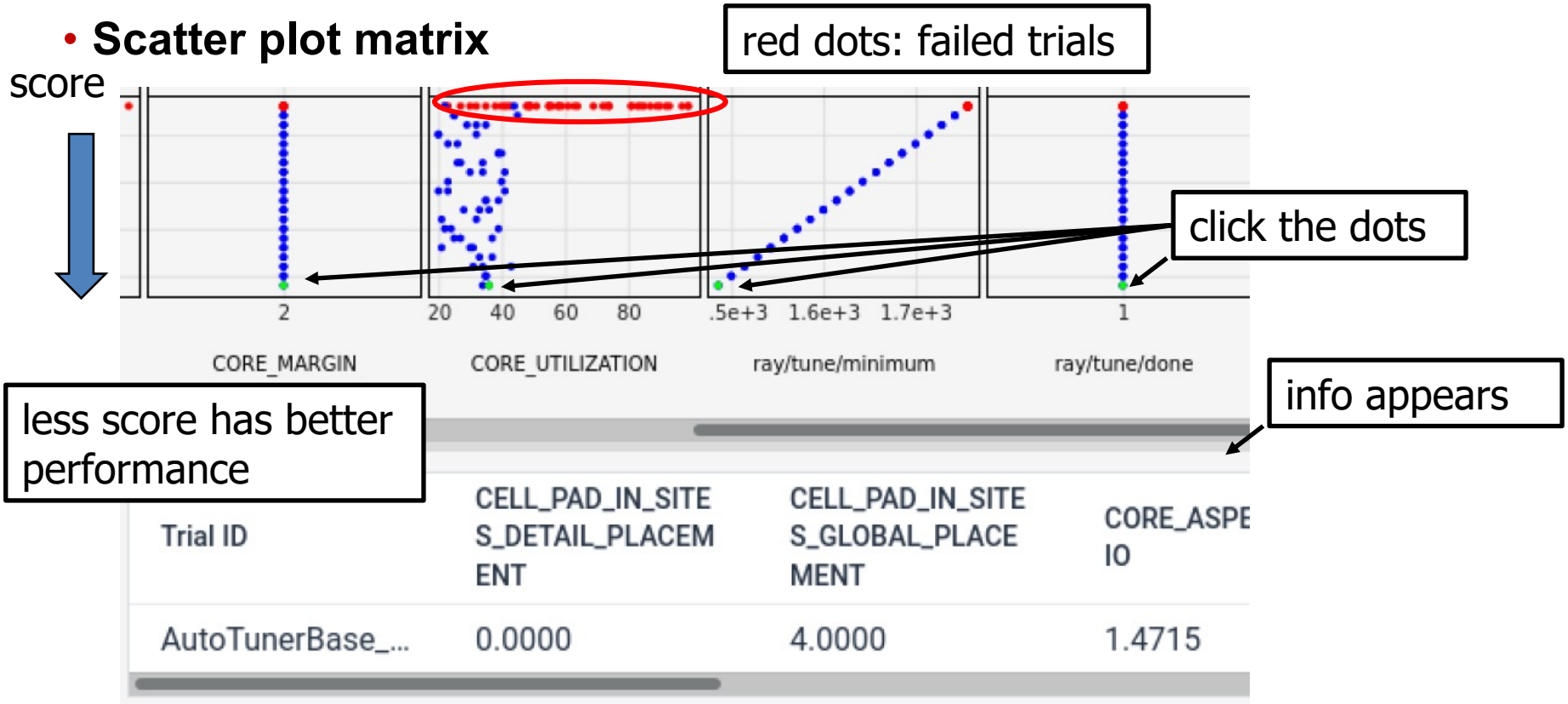
- **How to see the results**
  - **Scatter plot matrix**



(2) scoll down to find 'minimum'

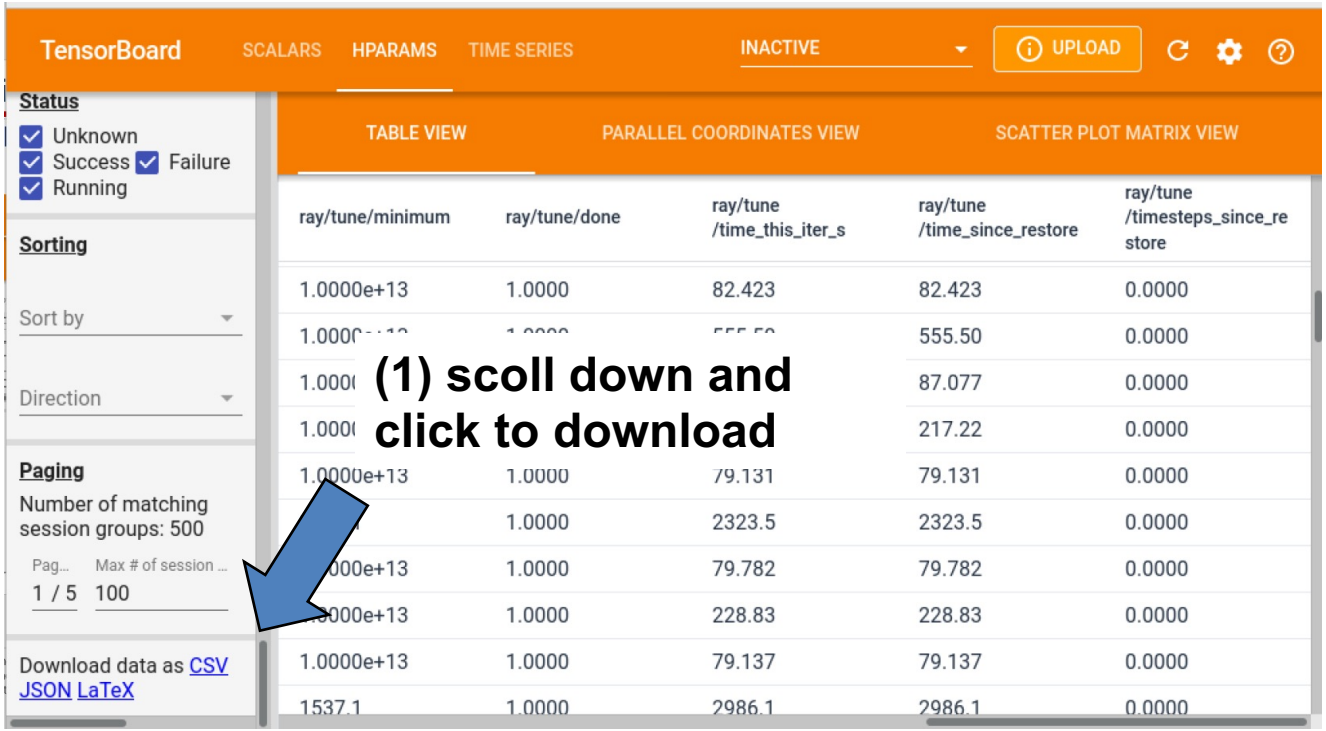
Demonstration – How to use GUI (Tensorflow GUI) (9)

- **How to see the results**
  - **Scatter plot matrix**



Demonstration – How to use GUI (Tensorflow GUI) (10)

- **How to see the results**
  - **Download CSV, JSON, LaTex**



Experiments and Results

- Experimental Setup
  - Framework is written in Python 3.8
  - Machine: Google Cloud Platform m1-ultramem-160. 160 2.2GHz vCPUs, 3.8TB memory
  - Platform**: ASAP 7nm, Skywater 130nm HD**
  - Design: **aes, ibex, jpeg**
  - Coefficient Setting (C_{power}, $C_{perform}$, C_{area})
    - **power$_{opt}$**: (10000, 100, 100)
    - **perform$_{opt}$**: (100, 10000, 100)
    - **area$_{opt}$**: (100, 100, 10000)
    - **uniform$_{opt}$**: (100, 100, 100)
  - All studies are parallelized to 40 concurrent OpenROAD jobs, with each job being assigned to 4 cores.

Experiments and Results

- Experimental Setup
  - Tunable tool and design parameters

Parameters	Description	Type	Range
_SDC_CLK_PERIOD	Target clock period (ns)	float	[T_{min}, T_{max}]
CORE_UTILIZATION	Target core utilization (%)	int	[20, 99]
CORE_ASPECT_RATIO	Floorplan aspect ratio	float	[0.1, 2.0]
CELL_PAD_IN_SITES_GLOBAL _PLACEMENT	Cell padding for global placement (site)	int	[0, 4]
CELL_PAD_IN_SITES_DETAIL _PLACEMEN	Cell padding for detailed placement (site)	int	[0, 4]
_FR_LAYER_ADJUST	Layer resource adjustment for global routing (%)	float	[0.1, 0.7]
PLACE_DENSITY_LB_ADDON	Additional lower bound increase of the target local global placement density (%)	float	[0.00, 0.99]
_SYNTH_FLATTEN	Design hierarchy flattening	int	[0, 1]
_PINS_DISTANCE	Minimum IO pad distance (#tracks)	int	[1, 3]
CTS_CLUSTER_SIZE	Target CTS sink cluster size	int	[10, 40]
CTS_CLUSTER_DIAMETER	Target CTS sink cluster diameter (um)	int	[80, 12]

Experiments and Results – Search Algorithm Comparison

- Coefficient setting: **perform$_{opt}$**
- Walltime constraint = 63966s (~17.8 hrs)
- Tuning scheduler: asynchronous scheduler

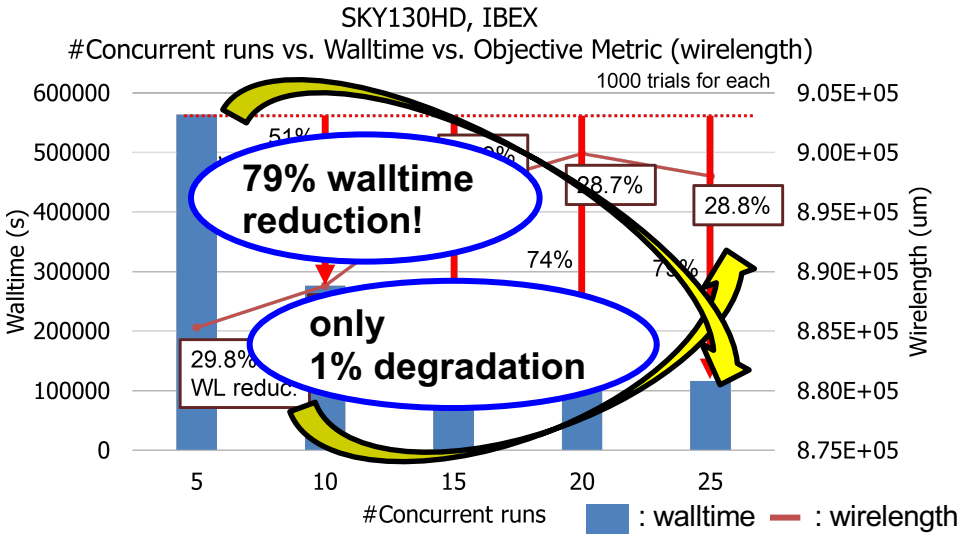
Algorithm	#trials	effCP (ns)	Improv.	Util (%)	Power	Improv.	Score
Reference		17.775		20	0.0523		1020000
PBT	1000	15.1621	15%	37	0.0143	73%	863611
HyperOpt	742	**14.9405**	**16%**	40	0.0286	45%	853503
Ax	478	16.0395	10%	26	0.0172	67%	914902
Optuna	1493	**14.8693**	**16%**	42	0.0278	47%	849094
Nevergrad	2925	15.2311	14%	18	0.0194	63%	870843

- **Nevergrad** have numerous trials by actively exploring loose hyperparameters or infeasible sets.
- **HyperOpt** and **Optuna** shows better (i.e., smaller) final scores
- However, appropriate search algorithm may vary depending on the design difficulty, experimental setup, CAD tools and parallelization!

Experiments and Results – Exploration vs. Exploitation

- High parallelization may cause performance degradation!
  - For example, if we use N concurrent jobs for N trials, walltime is minimized but this is the same as random ("multi-start") search (no iteration, no exploitation)

- Objective function: wirelength minimization
- #trials = 1000
- By parallelization,
  - **Reduces the total walltime** (left y-axis) up to 79%
  - **Worse performance** (wirelength) (right y-axis), but only by 1% degradation
- Why? → With asynchronous scheduler:
  - Walltime is proportional to $\frac{1}{\#concurrent\_jobs}$,
  - Performance degradation is almost proportional to $\frac{\#concurrent\_jobs}{\#trials}$



If the number of trials is large enough, active parallelization significantly reduces the total latency of autotuning

Experiments and Results – Automatic PPA Push

- 1000 trials for each setting. The table shows two of the three designs tested.
- For {SKY130, ASAP7} respectively, we achieve on average:
- {41%, 32%} power consumption reduction with power setting
- {21%, 18%} performance improvement with performance setting
- {68%, 58%} area reduction with area setting
- Comparison to OpenROAD-flow-scripts design-platform default in master branch
- Walltime: all < 2 days

Platform	Design	Setting	effCP (ns)	Improv.	Util (%)	Improv.	Power (W)	Improv.
SKY130	aes	ref	4.554	0%	19	0%	0.0838	0%
		power	4.373	4%	23	17%	**0.0569**	**32%**
		perf	**3.491**	**23%**	22	14%	0.0942	-12%
		area	3.763	17%	**42**	**55%**	0.0558	33%
		uniform	3.38	26%	35	46%	0.0612	27%
	jpeg	ref	9.277	0%	20	0%	0.0905	0%
		power	7.781	16%	37	46%	**0.0711**	**21%**
		perf	**7.259**	**22%**	31	35%	0.135	-49%
		area	12.186	-31%	**89**	**78%**	0.101	-12%
		uniform	7.88	15%	84	76%	0.08	12%
Platform	**Design**	**Setting**	**effCP (ps)**	**Improv.**	**Util (%)**	**Improv.**	**Power (W)**	**Improv.**
ASAP7	aes	ref	497.630	0%	29	0%	0.0170	0%
		power	445.819	10%	64	55%	**0.0098**	**42%**
		perf.	**408.012**	**18%**	52	44%	0.0325	-91%
		area	462.413	7%	**70**	**59%**	0.0154	9%
		uniform	443.146	11%	60	52%	0.0102	40%
	jpeg	ref	1039.200	0%	30	0%	0.0348	0%
		power	1021.997	2%	58	48%	**0.0340**	**2%**
		perf.	**838.097**	**19%**	59	49%	0.0520	-49%
		area	1066.411	-3%	**68**	**56%**	0.0366	-5%
		uniform	945.232	9%	64	53%	0.0379	-9%

Summary

- **METRICS2.1 infrastructure (Tutorial Part 5)**
  - An initiative of the IEEE CEDA Design Automation Technical Committee (DATC)
  - Enables standardized metrics collection and design process recording
  - We provide archives, reproducible config files and ML applications in the form of Jupyter notebooks
- **Flow Tuning (AutoTuner) (Tutorial Part 6)**
  - Open-source design flow autotuning framework
  - Supports synchronous/asynchronous parallelization, various search algorithms, and metrics collection using METRICS2.1
- **We invite you for collaboration!**
  - METRICS2.1
    - https://github.com/ieee-ceda-datc/datc-rdf-Metrics4ML
  - Flow Tuning (AutoTuner)
    - https://github.com/ieee-ceda-datc/datc-rdf-flow-tuner

THANK YOU !

We thank the OpenROAD team and the IEEE CEDA Design Automation Technical Committee for many helpful interactions.

This work is supported in part by the U.S. National Science Foundation (CCF-2112665) and the U.S. Defense Advanced Research Projects Agency (HR0011-18-2-0032).
https://theopenroadproject.org https://tilos.ai https://vlsicad.ucsd.edu

Part 7

Summary and Future Work

Jinwook Jung and Andrew B. Kahng

Summary and Future Work

- **Coming full circle: EDA = Optimization**
- **Futures in Learning + Optimization**

EDA is Optimization

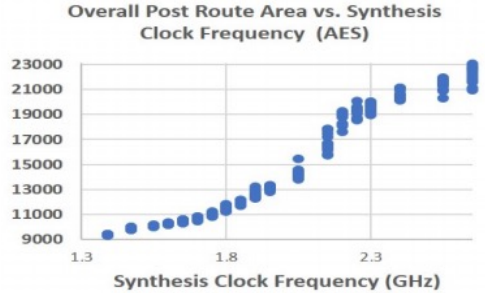
- **EDA is about optimizations and algorithms**
  - High stakes: performance, power, design closure
- **Discrete, combinatorial formulations at huge scale**
  - **Optimizations:** ILP, MCF, QAP, SAT/SMT, LR, …
  - **Algorithms:** min-cost flow, high-dim DP, …
- **But need an answer overnight**
- Reality under the hood: **metaheuristics**
  - Annealing, multi-start, PSO, ripup-reroute, greed, …
  - Convenient objectives, customer-/tech-specific tuning, …
  - *… which comes at a cost*

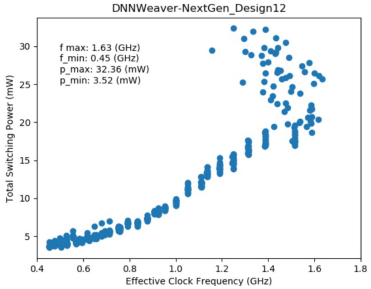
Never Enough Time: Heuristics and Chaos

- **"CAD tools are chaos machines"**

-- Ward Vercruysse, Sun UltraSPARC III CAD manager, **Physical Design Workshop 1996**

  - Push harder on a tool that is made up of heuristics stacked on top of heuristics → result becomes less predictable
  - Change initial conditions slightly → outcome can change a lot





- Recurring theme in my group…
  - ISQED02 (**Noise**), ISQED10 (**Chaos**)
  - DAC18 WIP (**Multi-Armed Bandits**)

Measurement of Inherent Noise in EDA Tools *

Andrew B. Kahng † and Stefanus Mantik

† UCSD CSE and ECE Departments, La Jolla, CA 92093-0114
UCLA Computer Science Department, Los Angeles, CA 90095-1596
abk@ucsd.edu, stefanus@cs.ucla.edu

Methodology From Chaos in IC Implementation

Kwangok Jeong[1] and Andrew B. Kahng[1,2]
[1]ECE and [2]CSE Departments, University of California at San Diego, La Jolla, CA, USA
kjeong@vlsicad.ucsd.edu, abk@cs.ucsd.edu

A No-Human-in-the-Loop Methodology Toward Optimal Utilization of EDA Tools and Flows

Andrew B. Kahng, Shriram Kumar and Tushar Shah, UCSD

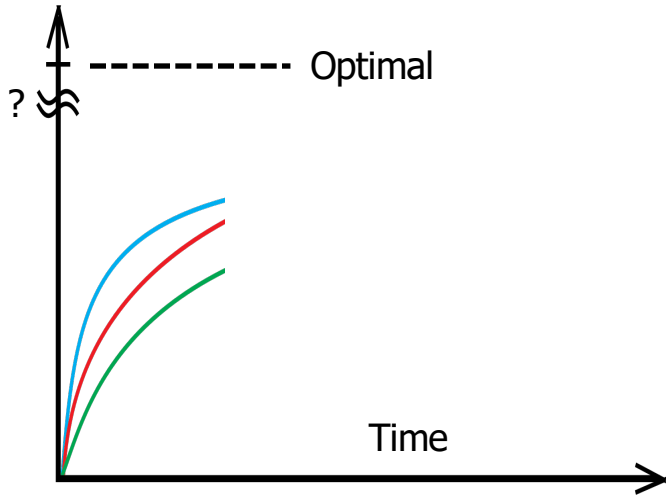
55 DESIGN AUTOMATION CONFERENCE
FROM CHIPS TO SYSTEMS – LEARN TODAY, CREATE TOMORROW

IC Design = "Multi-Armed Bandit" Problem

- **Multi Armed Bandit Problem (MAB): Given a slot machine with N arms, maximize total reward obtained using T "pulls" (iterations)**
  - Involves an explore-exploit tradeoff
    - Draw samples to learn model parameters (e.g., distributions of outcomes)
    - Simultaneously maximize reward
- **IC Design Problem**
  - Each "arm" = a target frequency.
  - Each "pull" = a run of the tool flow
- **Three well-known sampling strategies**
  - Thompson Sampling
  - ϵ-Greedy Sampling
  - Softmax Sampling
  - (+ Naïve: uniformly sample from all available arms)

Have We Lost Sight of the Suboptimality Gap?

- **Reality of optimization**
  - **Better, faster, cheaper – pick any two**
- **IC EDA: want all three at once**
  - **"Unfortunately, the runtime of …"**
- **But the world has changed**
  - **Automation, cloud, …**



Question: If you give your SP&R flow 3 extra days of runtime, would it know how to use this extra time?

Question: If you could run 10,000 copies of your P&R tool at the same time, what QOR improvement **should** you expect?

Generic Need: Re-Focus on Suboptimality (to get closer to Optimality!)

Suboptimality … in what sense?

- Need proper aiming points (which AI can help us **learn**)

Benchmarking

- "Real" benchmarks in EDA have been obfuscated, incomplete, non-vertical, old…
- "Artificial" benchmarks have tiptoed between realism, known optimal solution quality, scalability …
- Enough (30+ years of this) is enough → find a next level
  - E.g., "Underwriters Laboratories for IC Design Tools"

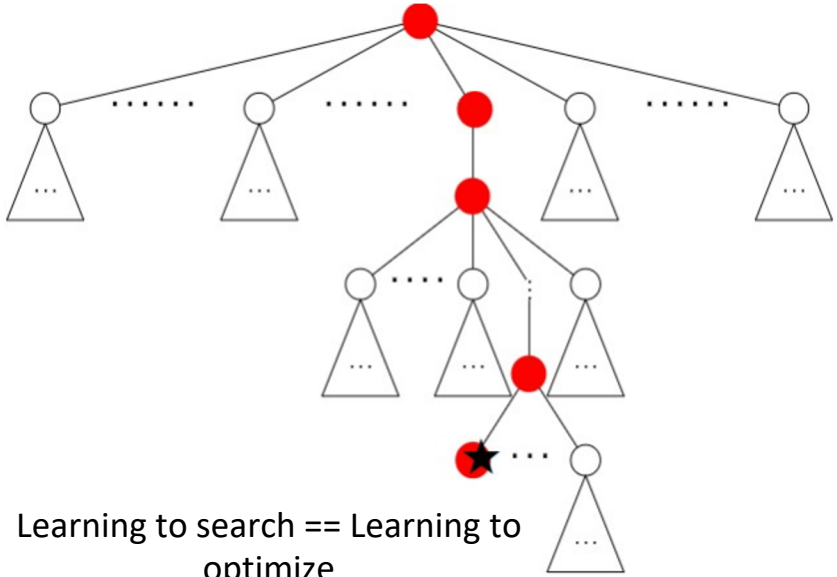
Comprehending modern compute resources *cloud, GPU, accelerators, …*

- EDA Optimization + Learning **naturally** live in the cloud
- **Most of today's EDA optimization algorithms are "EDA1.0" from the 1980s**
  - **→ Develop new, cloud-scalable "EDA2.0"**

Optimization Challenges

- **Why is optimization difficult?**
  - Combinatorial and high-dimensional
  - Many optimizers/solvers and tuning knobs
    - Greedy, tree-search, branch and bound
    - Gradient descent

- **EDA optimization adds further constraints**
  - Technology, libraries, PVT, RC
  - Tool (per stage)
  - Layout, timing constraints
  - Design rules
  - Tool parameter settings
  - Many more ….



Learning to search == Learning to optimize

How to automate efficient search?
Can we formalize learning problem?

Credit: S. Nath, NVIDIA

Learning to Optimize

Minimize $f(x_1,x_2,x_3, \ldots)$

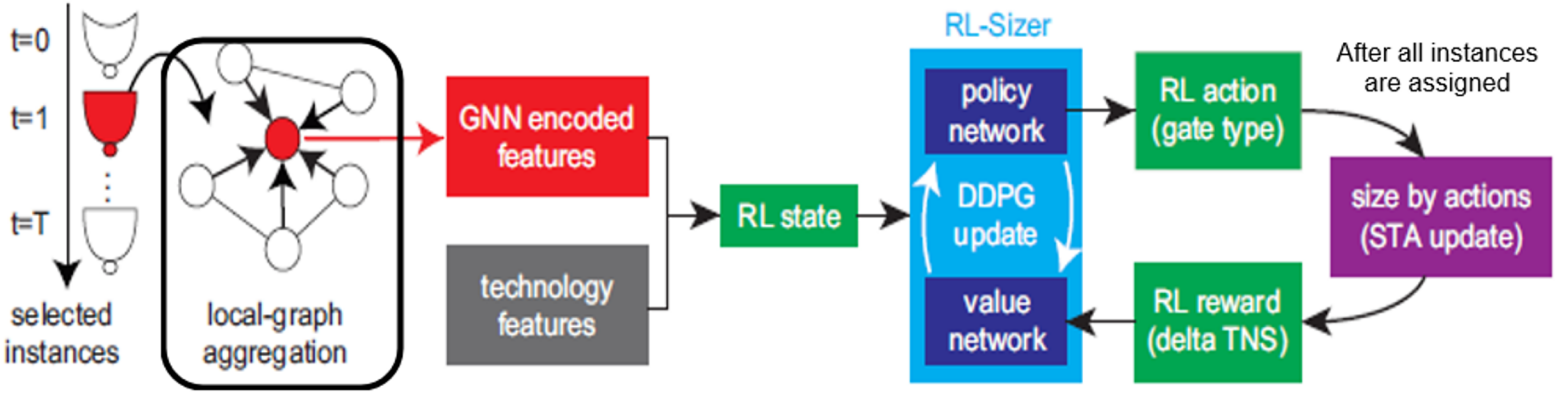
Subject to $g(x_1,x_2,x_3, \ldots)$

State: $x_1,x_2,\ldots,x_{i-1}$

Action: x_i

Reward: $f(x_1,x_2,x_3,\ldots)$, $g(x_1,x_2,x_3,\ldots)$
or $f`(x_1,x_2,\ldots,x_i)$, $g`(x_1,x_2,\ldots,x_i)$

- **Key questions**
  - **What** are we learning?
  - **Which** algorithm / model are we learning?
  - **How** are we learning?
- **Reinforcement learning is promising for sequential decision-making**
  - Sequential approach to select the best action with feedback (reward)
- **Recent RL-based gate sizing (DAC21) achieves up to *30% TNS improvement* w.r.t. commercial tool's sizer**



Credit: S. Nath, NVIDIA

Looking Ahead

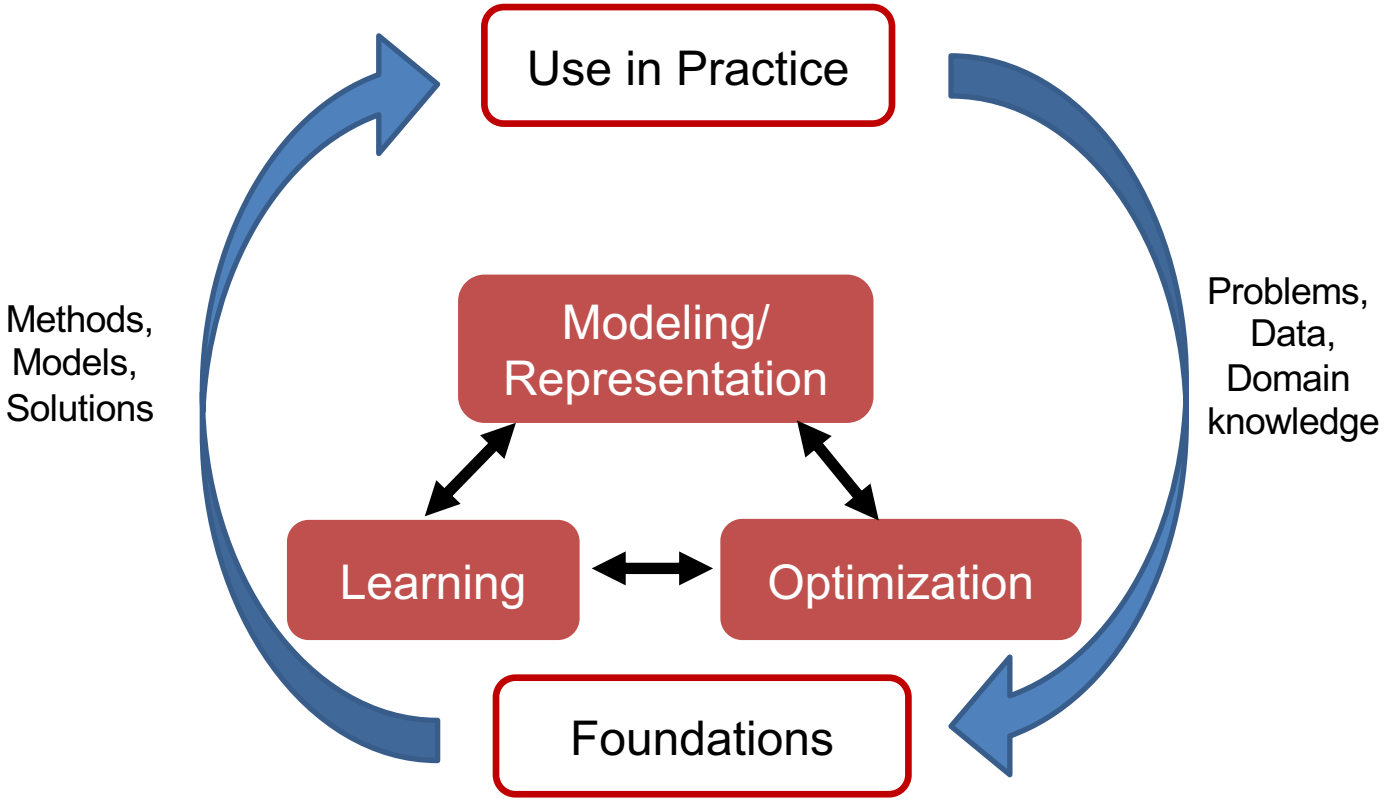
- **What are we learning? → Addressed today by majority of EDA works**
  - Base model vs. derived models (for task-specific learning such as congestion, global routing, etc.)
  - Extend to few-shot and transfer learning → largely lacking today
- **Which algorithm / model are we learning? → Largely restricted to hyperparameter tuning**
  - Ensemble, imitation learning that ***generalizes*** to multiple tasks for production use at industrial scale
- **How are we learning? → Not addressed today in the VLSI/EDA space**
  - Learn the learning rule / objective function best suited for a task
  - Opportunity for algorithm like RL to learn the objective function
  - General vs. specific model

Credit: S. Nath, NVIDIA

Learning + Optimization

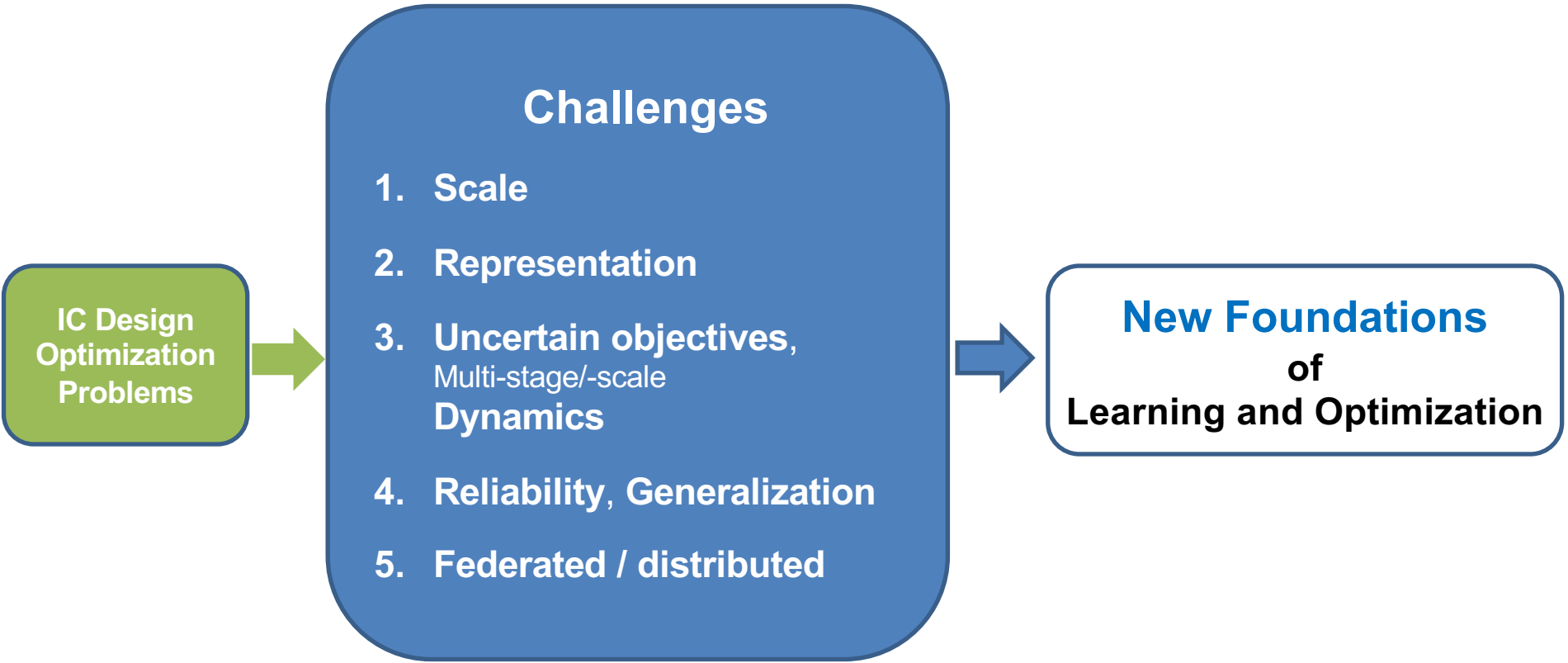
- **How Learning helps Optimization**
  - "Modeling and prediction": prune doomed runs early
  - "New cost-accuracy tradeoffs in analysis": less guardbanding
  - Value: more optimization within the available box of resources
  - Center of gravity for most of "ML in/around EDA" so far
- **How Optimization helps Learning**
  - Stochastic gradient descent, nonconvex optimization
  - Distributed/parallel
  - Meta-level: optimization of HW on which learning takes place
- **Virtuous cycle that brings Scaling (of CAD/EDA and IC design)**

Nexus of Theory and Practice



Credit: N. Vishnoi, S. Jegelka, D. Spielman, Y. Wang +

(This Will Keep Us Busy for … 5-10 Years?)



Credit: N. Vishnoi, S. Jegelka, D. Spielman, Y. Wang +

12

AI and Optimization: Key Directions to Watch

AI advances pose new challenges, provide new tools for optimization

Bridging Discrete and Continuous

Distributed, Parallel, and Federated

Optimization on Manifolds

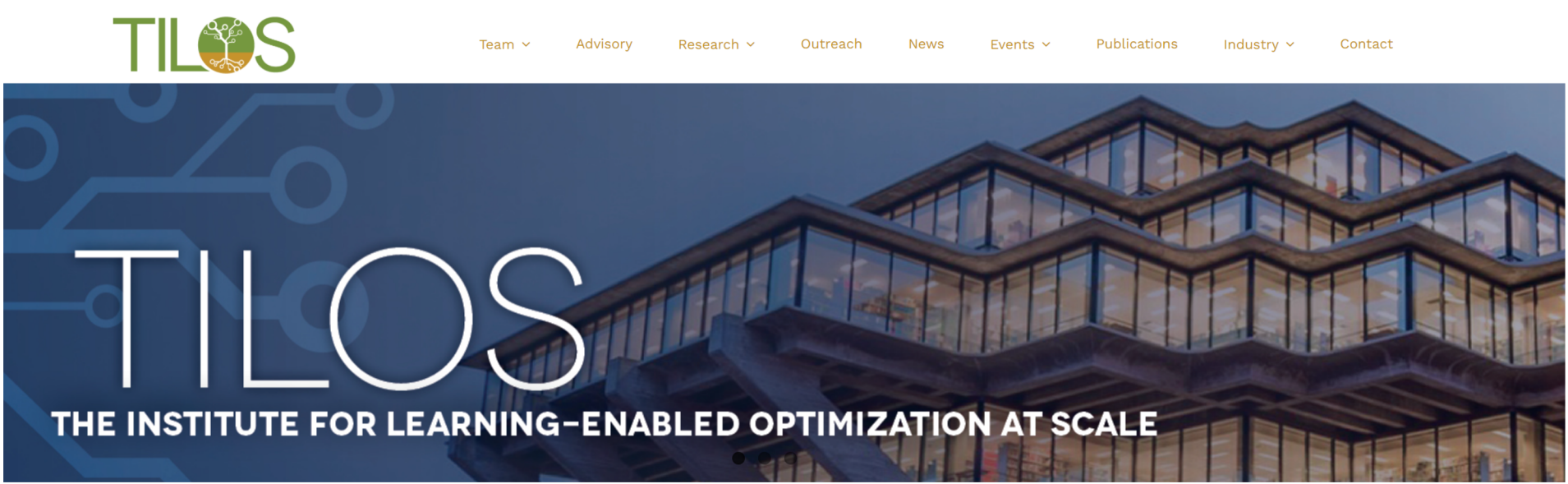
Dynamic Decisions under Uncertainty

Nonconvex Optimization in Deep Learning

New perspectives on classic problems → **watch this space!**

Credit: N. Vishnoi, S. Jegelka, D. Spielman, Y. Wang +

TILOS: An NSF AI Research Institute for Advances in Optimization



- Partially supported by Intel Corporation
- Launched on November 1, 2021

tilos.ai

THANK YOU !

- Research at UCSD is supported by NSF, DARPA, Qualcomm, Samsung, NXP, Mentor Graphics and the C-DEN center.

- Questions/Feedback: abk@eng.ucsd.edu

Summary of Tutorial 1

- We have presented IEEE CEDA DATC's efforts toward standard platform for ML-enabled EDA and IC design

- **Introduction** ..... Part 1
- **Challenges in ML-enabled EDA and IC design** ..... Part 2
- **DATC's Efforts on standard platform for ML-enabled EDA and IC design**
  - **Platform for EDA and IC Design**
    - DATC Robust Design Flow ..... Part 3
  - **Platform for "ML-Enabled" EDA and IC Design**
    - Large-scale data generation in cloud-native way ..... Part 4
    - METRICS 2.1: Toward standard metrics collection system ..... Part 5
    - AutoTuner: Autonomous flow tuning framework ..... Part 6
- **Summary and future work** ..... Part 7