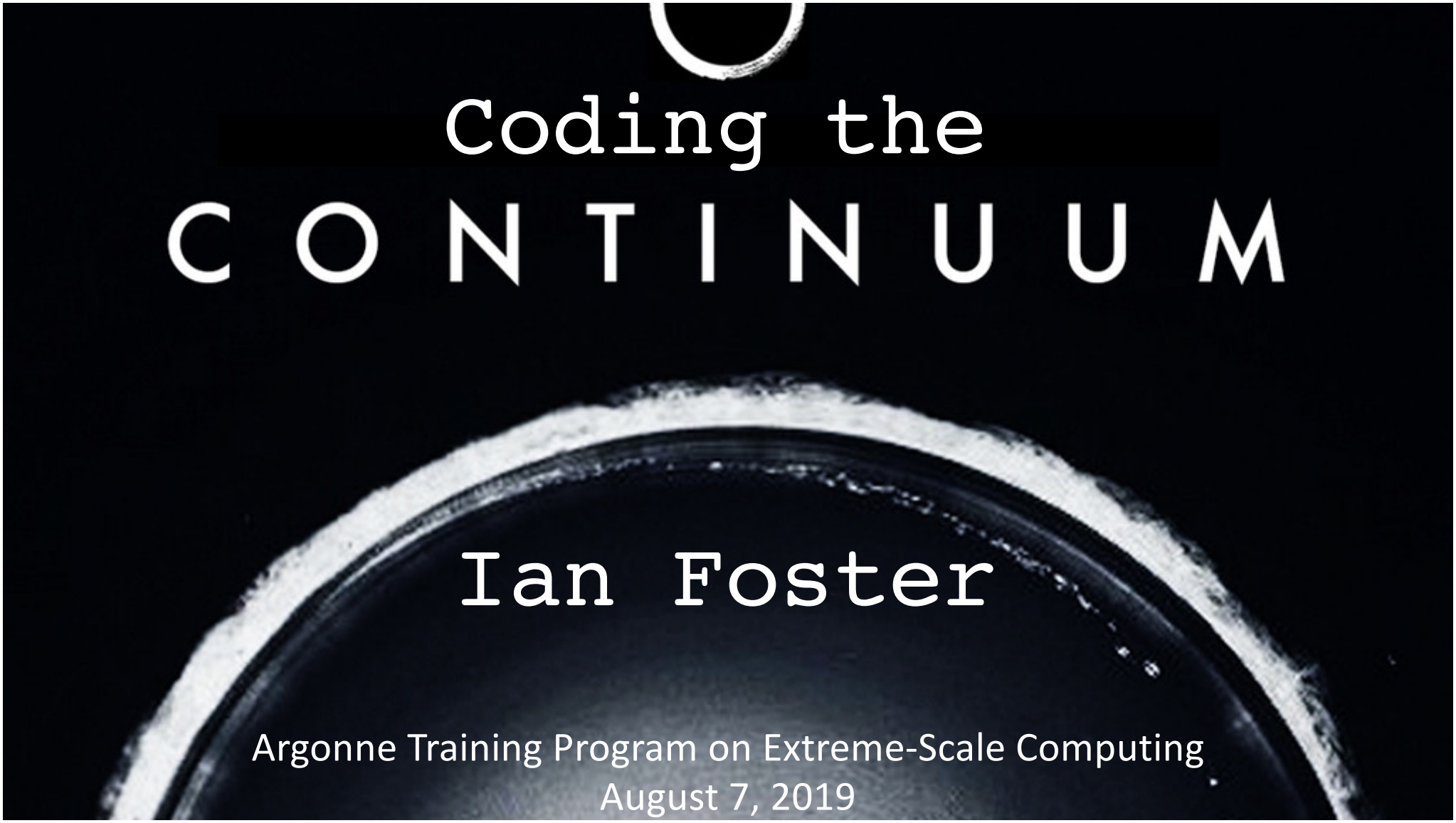

Coding the
CONTINUUM
Ian Foster
Argonne Training Program on Extreme-Scale Computing
August 7, 2019

Then (1980) Now (2019)

Supercomputing

1.6×10^8 flop/s (Cray-1) 1.5×10^{17} flop/s (Summit)

Then (1980)	Now (2019)
Supercomputing	
1.6×10^8 flop/s (Cray-1)	1.5×10^{17} flop/s (Summit)
Wide area networking	
10^4 bit/s (dial-up)	10^{11} bit/s (optical)

"When the network is as fast as the computer's internal links, the machine disintegrates across the net into a set of special-purpose appliances."

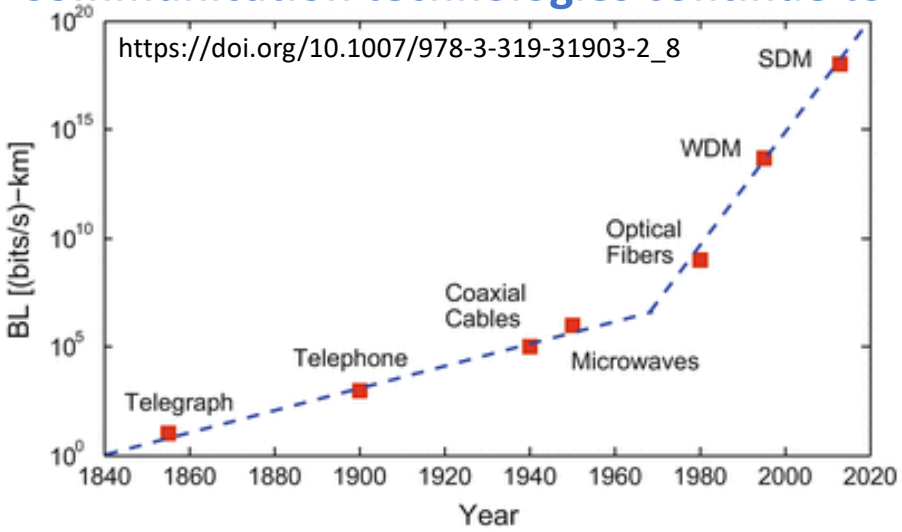
-- George Gilder, 2000

"When the network is as fast as the computer's internal links, the machine disintegrates across the net into a set of special-purpose appliances."

-- George Gilder, 2000

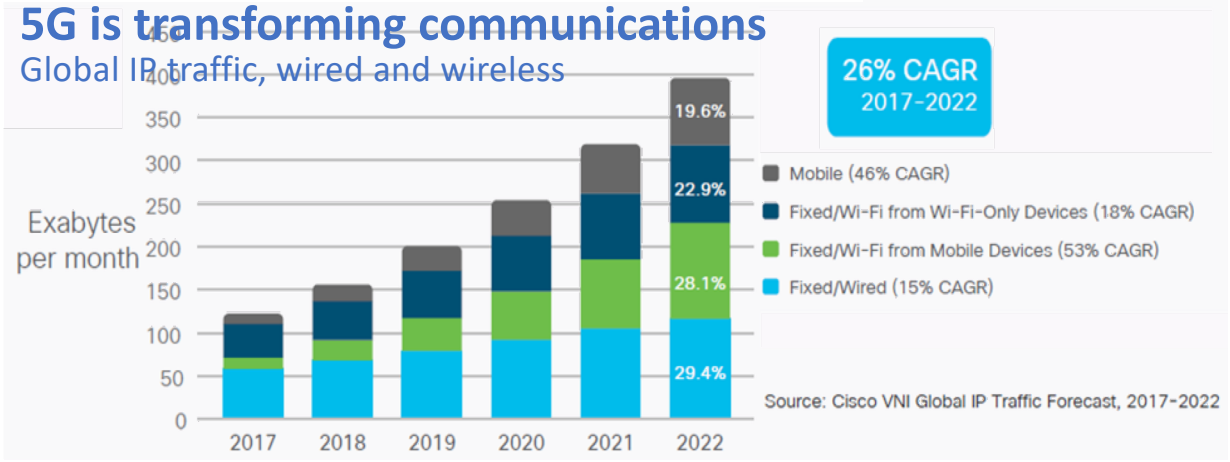
“network is as fast as the computer’s internal links”

Communication technologies continue to evolve



5G is transforming communications

Global IP traffic, wired and wireless



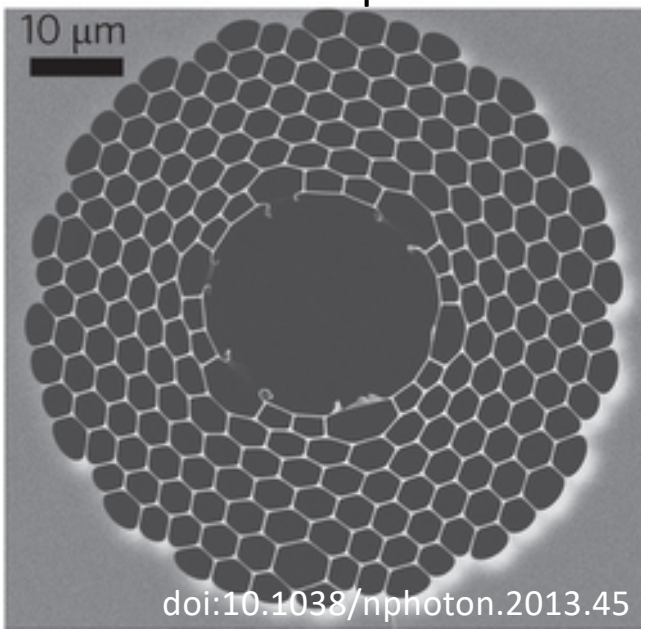
Innovation continues in the lab

Hollow core fiber:

99.7% speed of light

(1.46x faster than fiber)

73.7 terabits per second



We can compute anywhere!

But are we really free?

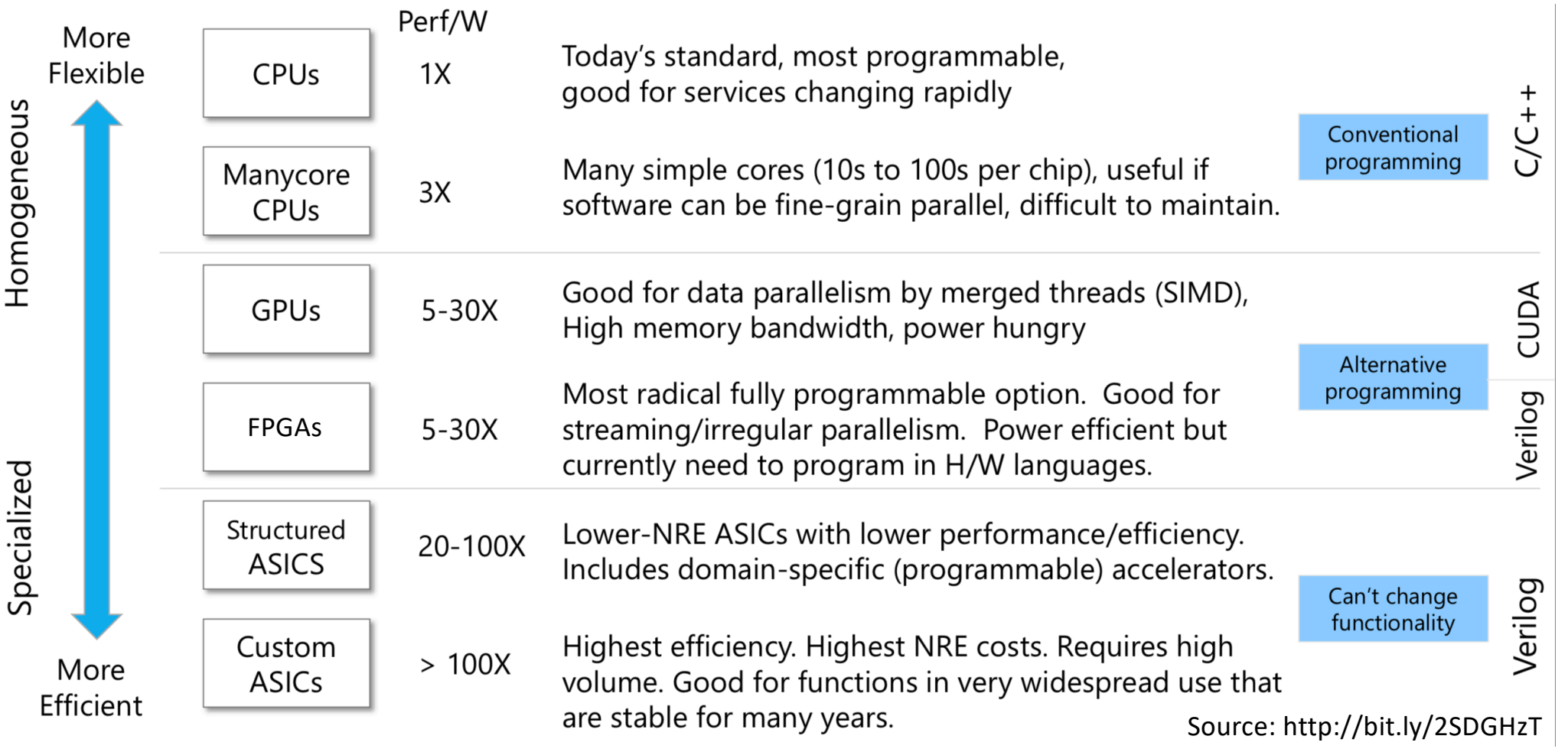
$$\text{Time} = T_{\text{compute}} + 2\,T_{\text{latency}}$$

*207 000 km/sec in optical fiber:
5×10^{-6} sec/km

"When the network is as fast as the computer's internal links, the machine disintegrates across the net into a set of special-purpose appliances."

-- George Gilder, 2001

“a set of special-purpose appliances”



Source: http://bit.ly/2SDGHzT

AI chip landscape

More on https://basicmi.github.io/AI-Chip/

Tech Giants/Systems

Google
Microsoft
aws
IBM
facebook
Apple
Tesla
HUAWEI
Baidu 百度
Alibaba Group 阿里巴巴集团
FUJITSU
NOKIA
TOSHIBA
Hewlett Packard Enterprise
DELL

IC Vender/Fabless

intel
SAMSUNG
NVIDIA
QUALCOMM
AMD
NXP
ST
XILINX
MEDIATEK
BROADCOM
MARVELL
Rockchip 瑞芯微电子

IP/Design Service

arm
SYNOPSYS
Imagination
cadence
CEVA
VeriSilicon
SiFive
ARTERIS IP
Alchip
GUC
FARADAY
eSilicon

Startup in China

Cambricon 寒武纪科技
地平线 Horizon Robotics
BITMAIN
intelliFusion 云天励飞
ChipIntelli
Think Force
Canaan
云知声 Unisound
Rokid
AISPEECH 思必驰 专注人性化的智能语音
肇观电子 NextVPU
Enflame
亿智科技
清微智能 TSING MICRO

Startup Worldwide

cerebras
WAVE COMPUTING
Graphcore
habana
thinci
SambaNova SYSTEMS
KALRAY
LIGHTELLIGENCE
HAILO Empowering Intelligence
Esperanto TECHNOLOGIES
Tenstorrent
MYTHIC
Preferred Networks
brainchip
PEZY Computing
GREENWAVES TECHNOLOGIES
AIMOTIVE
KONIKU
Tachyum
flexlogix
SYNTIANT
gyrfalcon technology
NOVUMIND

扫码访问AI芯片文章

Compiler

GLOW
tvm
NVIDIA TensorRT
ONNC
nGraph Compiler stack (Beta)
plaidML

Benchmarks

MLPerf
AI - Benchmark
AI Matrix.

“a set of special-purpose appliances”

“**Cloud computing 5x to 10x improved price point** [relative to Enterprise]”

— James Hamilton, http://bit.ly/2E78Wi1

Why?

- Improved utilization
- Economies of scale in operations
- More power efficient
- Optimized software

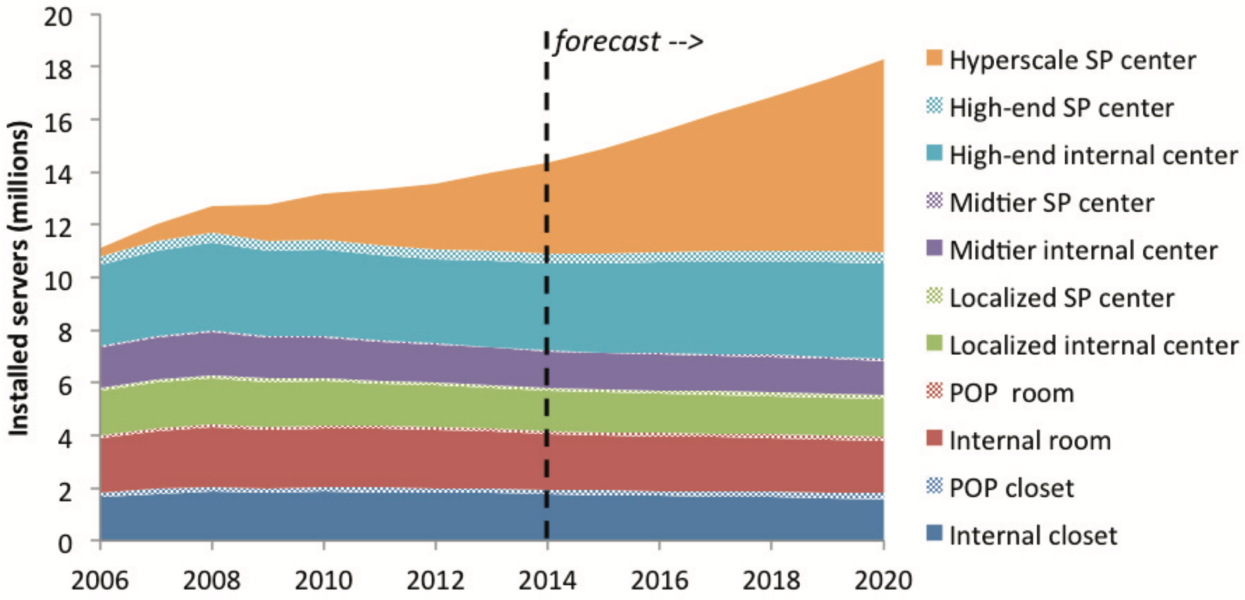
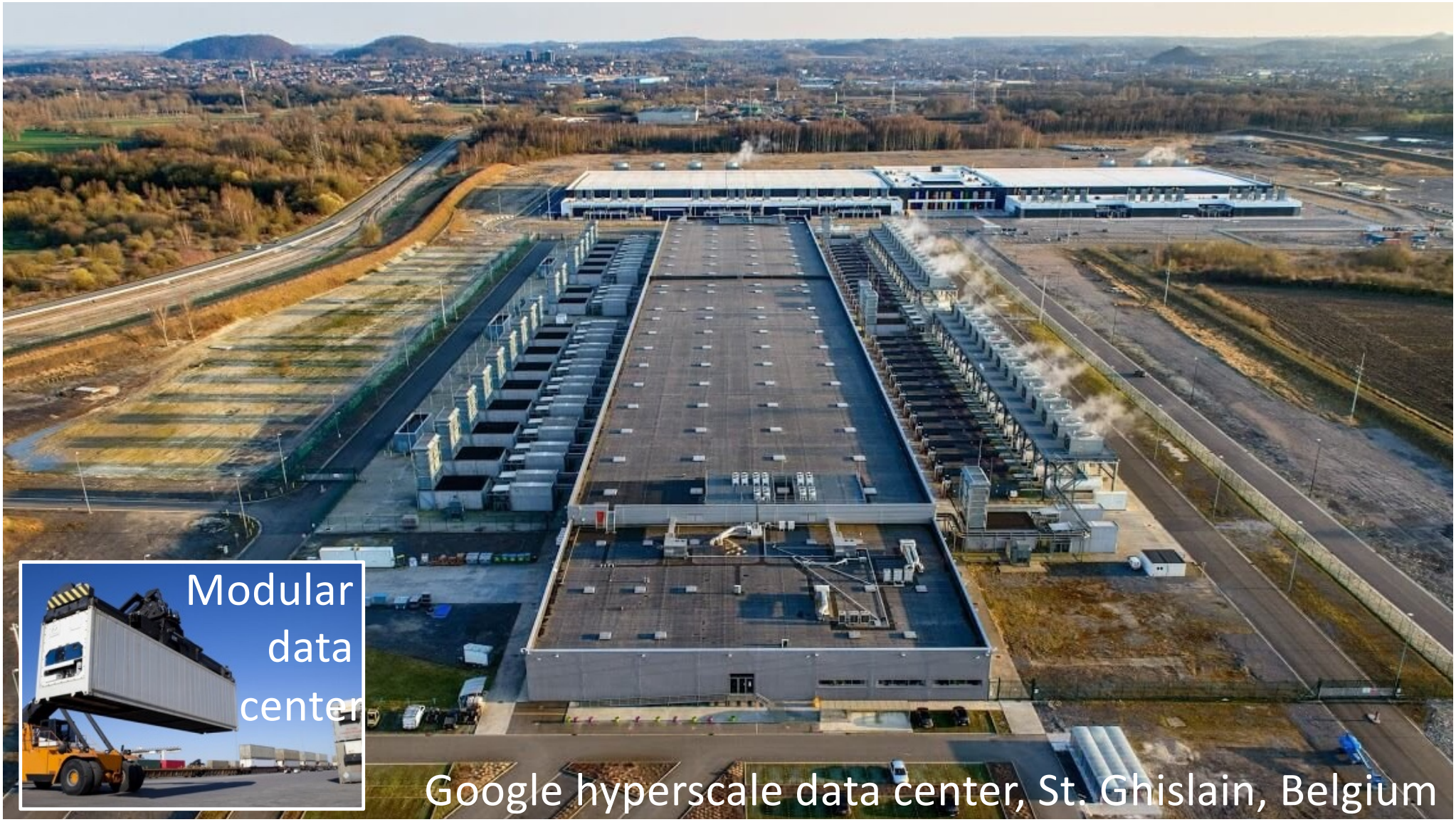


Figure 20. Total Server Installed Base by Data Center Space Category

LBNL-1005775


Modular data center
Google hyperscale data center, St. Ghislain, Belgium

Zero-carbon cloud: Reduce energy cost and energy carbon footprint to 0

Andrew Chien DOI 10.1109/IPDPS.2016.96

Energy Policy Institute's Fifth Annual Energy Policy Research Conference

Dr. Andrew A. Chien *is the William Eckhardt Professor in Computer Science and Director of the CERES Center for Unstoppable Computing at the University of Chicago and a Senior Computer Scientist at Argonne National Laboratory. From 2005 to 2010, Chien was Vice President of Research of Intel Corporation, and from 1998 to 2005, the SAIC Chair Professor in Computer Science and founder of the Center for Networked Systems at UCSD. His research has been recognized for excellence with numerous awards, and also supported by the NSF, DARPA, DOE, ONR, NASA, and industry. Dr. Chien currently serves as co-chair of the editorial board of the Communications of the ACM. From 1990 to 1998, he was a Professor of Computer Science at University of Illinois. Dr. Chien earned B.S., M.S., and Ph.D. degrees at the Massachusetts Institute of Technology, and is a Fellow of the ACM, IEEE, and AAAS.*

Dr. Rich Wolski *is a Professor of Computer Science at the University of California, Santa Barbara, and co-founder of Eucalyptus Systems Inc. Having received his M.S. and Ph.D. degrees from the University of California at Davis (while a research scientist at Lawrence Livermore National Laboratory) he has also held positions at the University of California, San Diego, and the University of Tennessee, the San Diego Supercomputer Center, and Lawrence Berkeley National Laboratory. Dr. Wolski has led several national-scale research efforts in the area of distributed systems and is the progenitor of the Eucalyptus open source cloud project.*

Fan Yang *is currently a third-year Ph.D. student in the Department of Computer Science, University of Chicago. She joined the Large-Scale System Group at UChicago in 2013. She has broad research interests in parallel and distributed systems, data-intensive computing, parallel programming and optimization, large-scale system software, and computer architecture. In the past year, she worked on graph computing and benchmarking of graph-processing systems. She currently focuses on green datacenter design and stranded power in renewable electricity generation. She received her B.Eng. in computer science from National University of Defense Technology, Changsha, Hunan Province, China, in 2012.*

This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Award DE-SC0008603 and Contract DE-AC02-06CH11357, as well as the National Science Foundation under Awards CNS-1405959, STCI-0751315, CNS-0905237, and CNS-1218808. The authors also gratefully acknowledge generous support from Hewlett-Packard, Keysight, Huawei, Nvidia, and the Seymour Goodman Foundation.

CrossMark

The Zero-Carbon Cloud: High-Value, Dispatchable Demand for Renewable Power Generators

Variability is an ongoing challenge to growth of large-scale renewable power generation, posing challenges for the power grid and ambitious renewable portfolio standards. The authors propose Zero-Carbon Cloud (ZCCloud), a new high-value, dispatchable demand for renewables that improves their economic viability. Initial studies show that ZCCloud can create high-value computing resources with payback periods of just a few years.

Andrew A. Chien, Richard Wolski and Fan Yang

I. Rising Renewable Power Standards: The Stranded Power Opportunity

Generation costs for renewable power are an ongoing challenge to growing adoption of higher targets for renewable portfolio standards (RPS). Among leading RPS states, California produced 20 percent of its power from renewable sources in 2010 and has an ambitious target of 33 percent by 2020, and even Midwestern states such as Illinois[1] have adopted RPS targets that increase renewables in the electrical utilities fuel mix from 10 percent in 2015 to 25 percent in 2025. The United States federal government has established a goal of 20 percent by 2020 for all federal agencies, a dramatic increase

  http://dx.doi.org/10.1016/j.tej.2015.09.010

The performance landscape becomes peculiar

A program can run on two computers

- On C_1, it takes 0.01 seconds
- On C_2, it takes 0.005 seconds

Which is faster?

The performance landscape becomes peculiar

A program can run on two computers

On C_1, it takes 0.01 seconds

On C_2, it takes 0.005 seconds

Which is faster?

The answer depends on their location.

Say C_1 is adjacent and C_2 is 500 km distant

$t(C_1) = T_1 = 0.01$ sec

$t(C_2) = T_2 + 2 \times 500 \times 5 \times 10^{-6} = 0.01$ sec

The performance landscape becomes peculiar

A program can run on two computers

On C_1, it takes 0.01 seconds

On C_2, it takes 0.005 seconds

Which is faster?

The answer depends on their location.

Say C_1 is adjacent and C_2 is 500 km distant

$t(C_1) = T_1 = 0.01$ sec

$t(C_2) = T_2 + 2 \times 500 \times 5 \times 10^{-6} = 0.01$ sec

The apparent **speed** of a computer depends on its **location**;
the apparent **location** of a computer depends on its **speed**

Continuum

A set of elements such that between any two of them there is a third element [dictionary.com]

For example, the **computing continuum:**

IoT/Edge — **Fog** — **HPC/Cloud**

Size	Nano	Micro	Milli	Server	Fog	Campus	Facility
Example	Adafruit Trinket	Particle.io Boron	Array of Things	Linux Box	Co-located Blades	1000-node cluster	Datacenter
Memory	0.5K	256K	8GB	32GB	256G	32TB	16PB
Network	BLE	WiFi/LTE	WiFi/LTE	1 GigE	10GigE	40GigE	N*100GigE
Cost	$5	$30	$600	$3K	$50K	$2M	$1000M

Count = 10^9
Size = 10^1

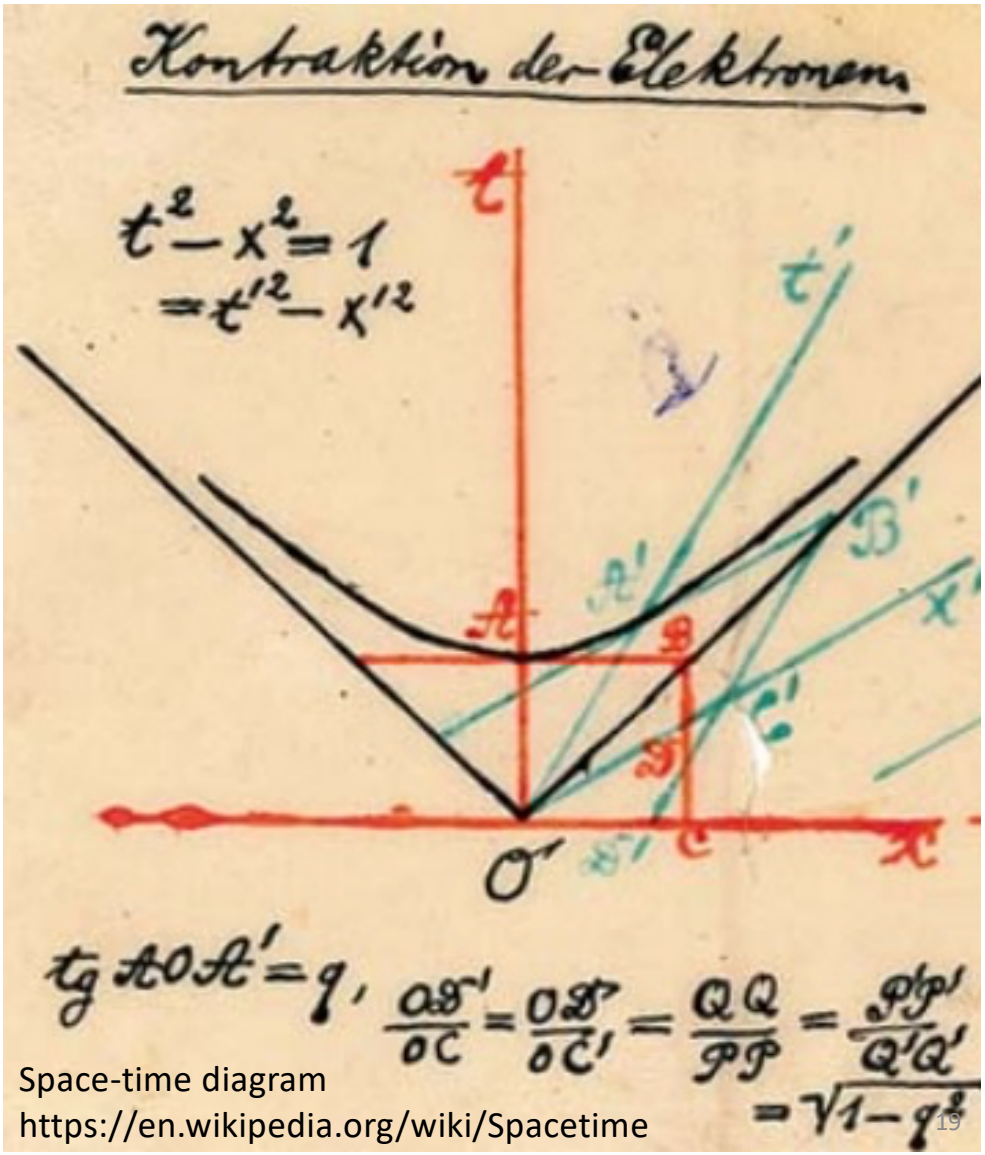
Count = 10^1
Size = 10^9

Credit: Pete Beckman, beckman@anl.gov

The
space-time continuum

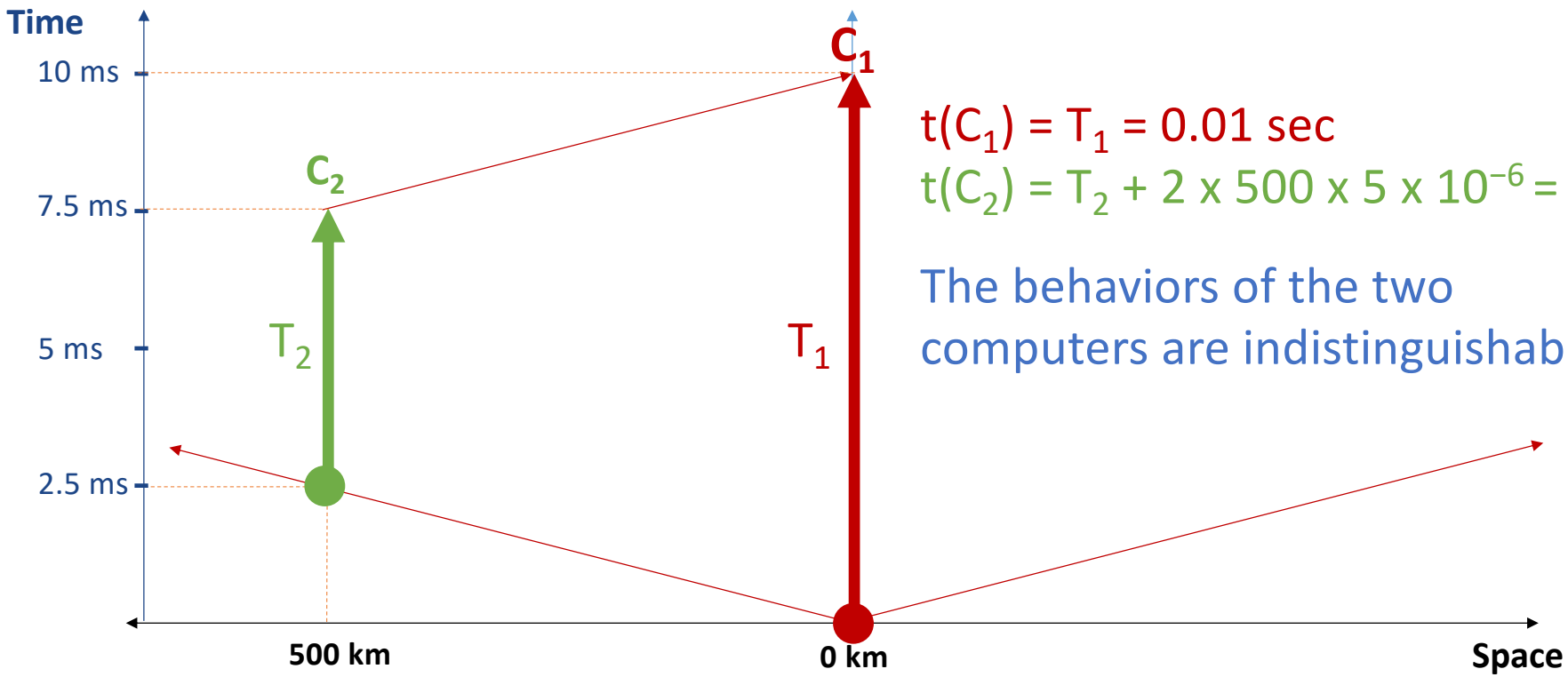
"space by itself, and time by itself, are doomed to fade away into mere shadows, and only a kind of union of the two will preserve an independent reality …"

H. Minkowski, 1908



Space-time diagram
https://en.wikipedia.org/wiki/Spacetime

The spacetime continuum in computational systems

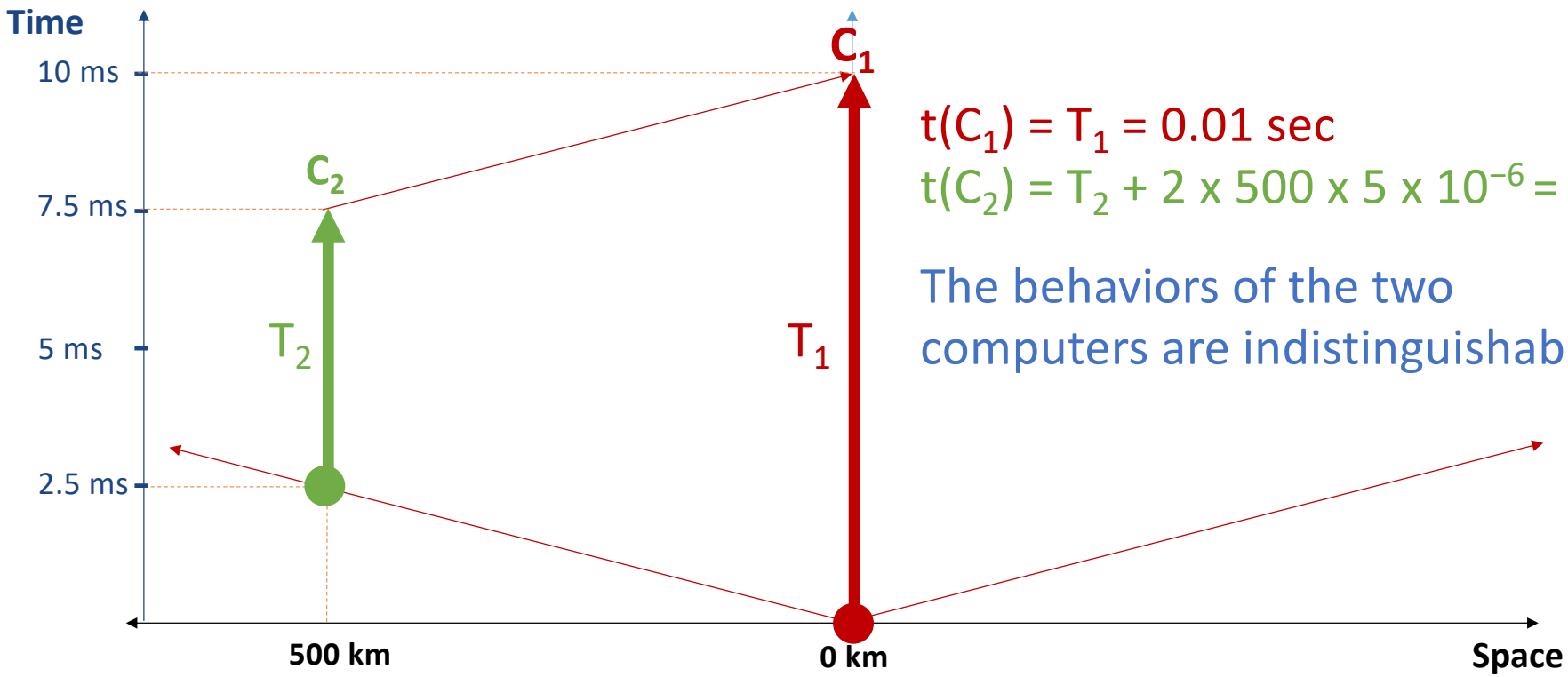


$t(C_1) = T_1 = 0.01$ sec

$t(C_2) = T_2 + 2 \times 500 \times 5 \times 10^{-6} = 0.01$ sec

The behaviors of the two computers are indistinguishable

The spacetime continuum in computational systems



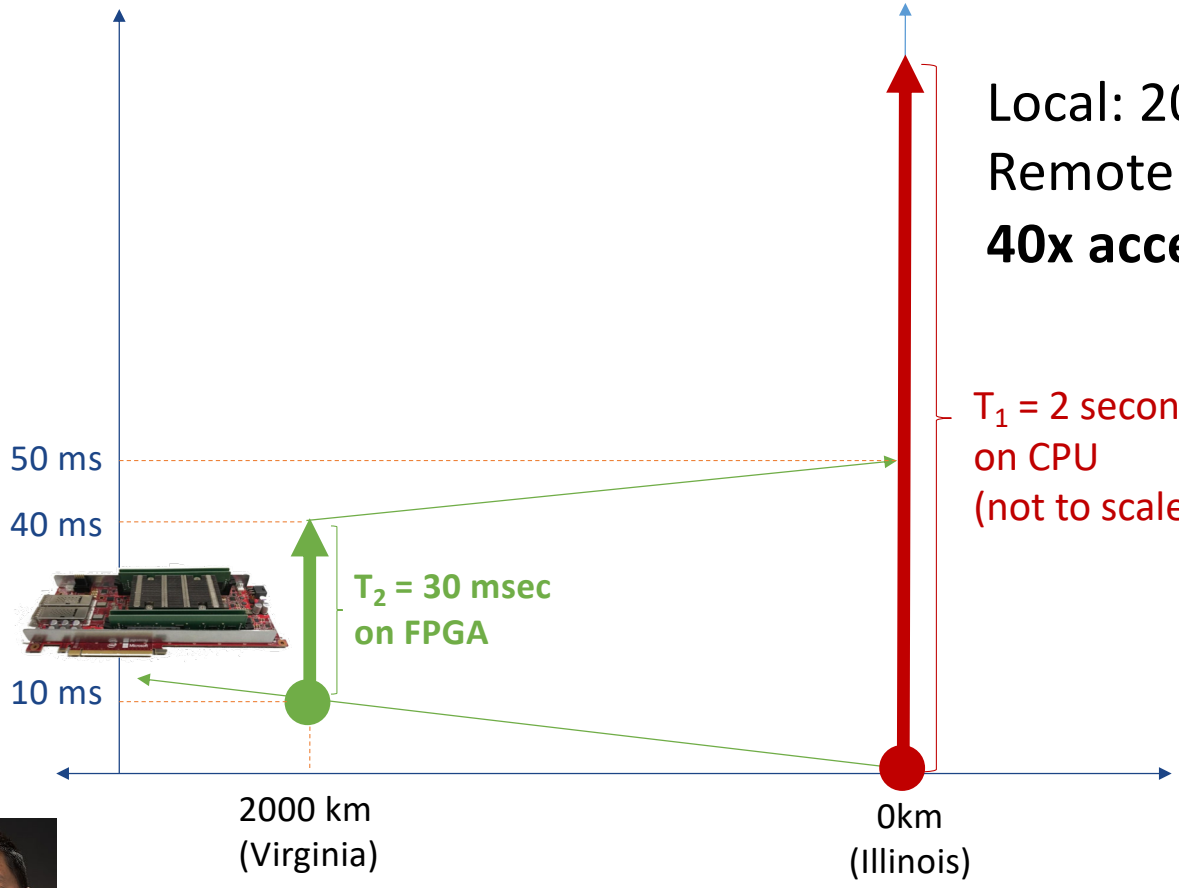
$t(C_1) = T_1 = 0.01$ sec

$t(C_2) = T_2 + 2 \times 500 \times 5 \times 10^{-6} = 0.01$ sec

The behaviors of the two computers are indistinguishable

Misquoting Minkowski: "*Henceforth, **location** for itself, and **speed** for itself shall completely reduce to a mere shadow, and only some sort of union of the two shall preserve independence*."

A real example: High energy physics trigger analysis



Local: 2000 msec

Remote: 30 + 10 + 10 = 50 msec

40x acceleration

Nhan Tran, FermiLab, et al. arXiv:1904.08986

Reasoning about the computing continuum (a) Assumptions

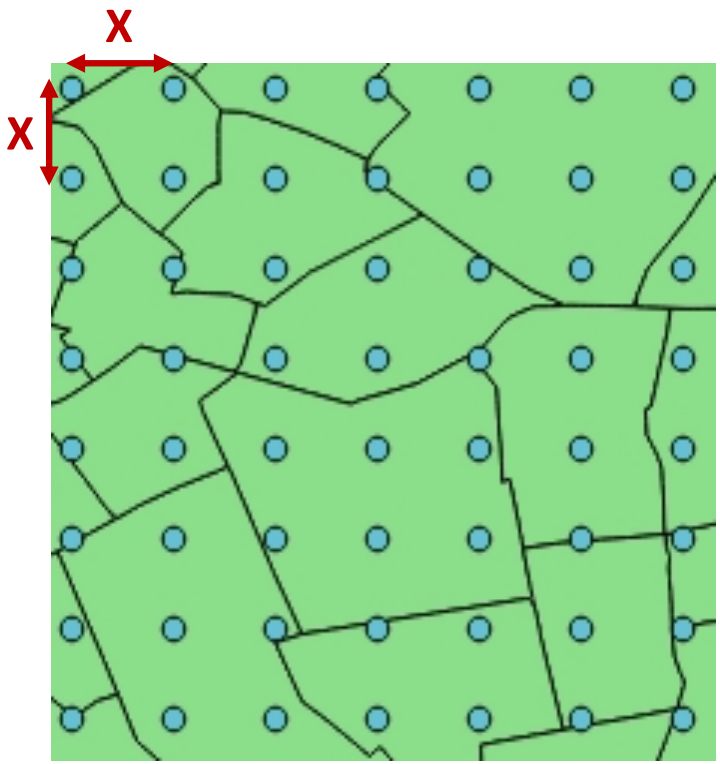
A1: N consumers, distributed uniformly, X secs apart

A2: Each consumer requests compute units at 1 Hz

A3: Infinite bandwidth: i.e., only compute time and latency are of concern

A4: An individual computer takes T secs to complete a compute unit

A5: A compute center containing Z computers is faster by a factor of $\sqrt{Z}$



Reasoning about the computing continuum (a) Assumptions

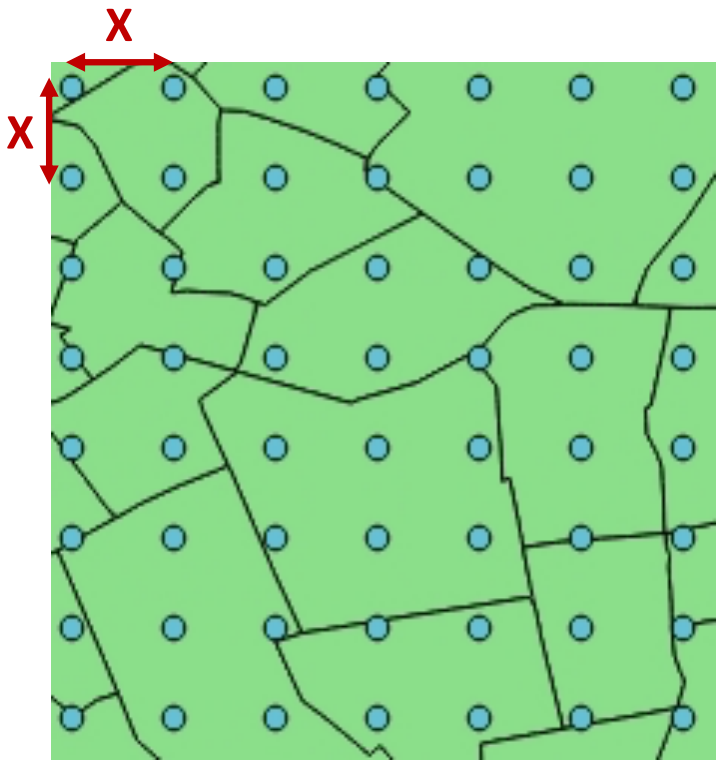
A1: N consumers, distributed uniformly, X secs apart

A2: Each consumer requests compute units at 1 Hz

A3: Infinite bandwidth: i.e., only compute time and latency are of concern

A4: An individual computer takes T secs to complete a compute unit

A5: A compute center containing Z computers is faster by a factor of $\sqrt{Z}$ *



* Grosch's law (1953): computer power rises by the square of the price

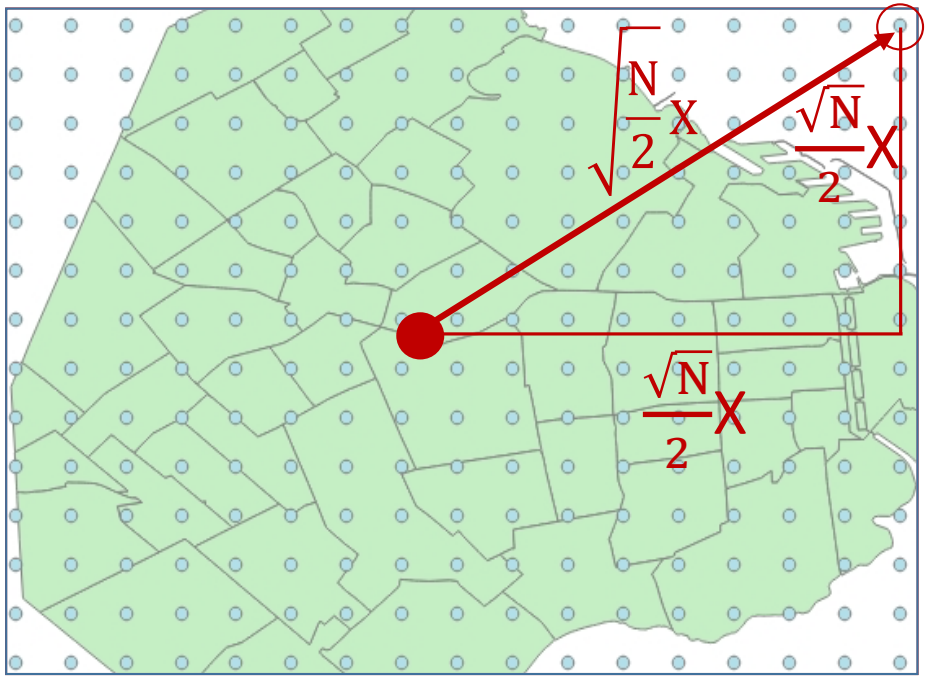
Reasoning about the computing continuum
(b) Without response time bounds

Max time is:

A) 1 local computer per consumer:

$t = T$

B) Center with N computers: $t = \frac{T}{\sqrt{N}} + 2\sqrt{\frac{N}{2}}\,X$



Reasoning about the computing continuum (b) Without response time bounds

Max time is:

E.g., N = 100, T=0.01, X=0.0001

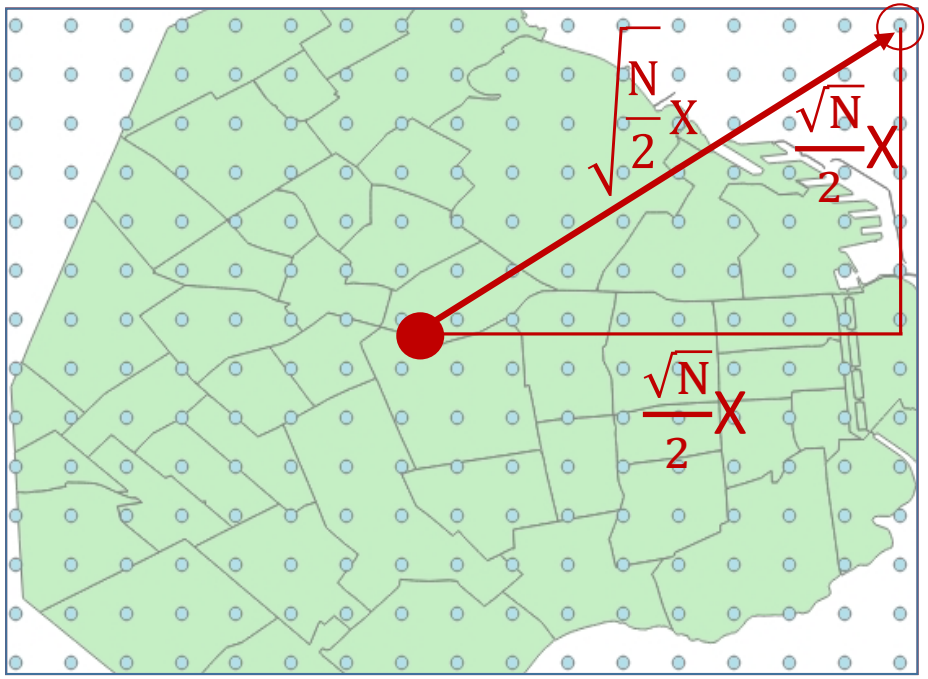
A) 1 local computer per consumer:

t = T = **0.01 s**

B) Center with N computers: $t = \frac{T}{\sqrt{N}} + 2\sqrt{\frac{N}{2}}\,X$

$$= \frac{0.01}{10} + 2\sqrt{\frac{100}{2}}\,0.0001$$

= 0.001 + 0.00142 = **0.00242 s**



Reasoning about the computing continuum (b) Without response time bounds

Max time is:

E.g., N = 100, T=0.01, X=0.0001

A) 1 local computer per consumer:

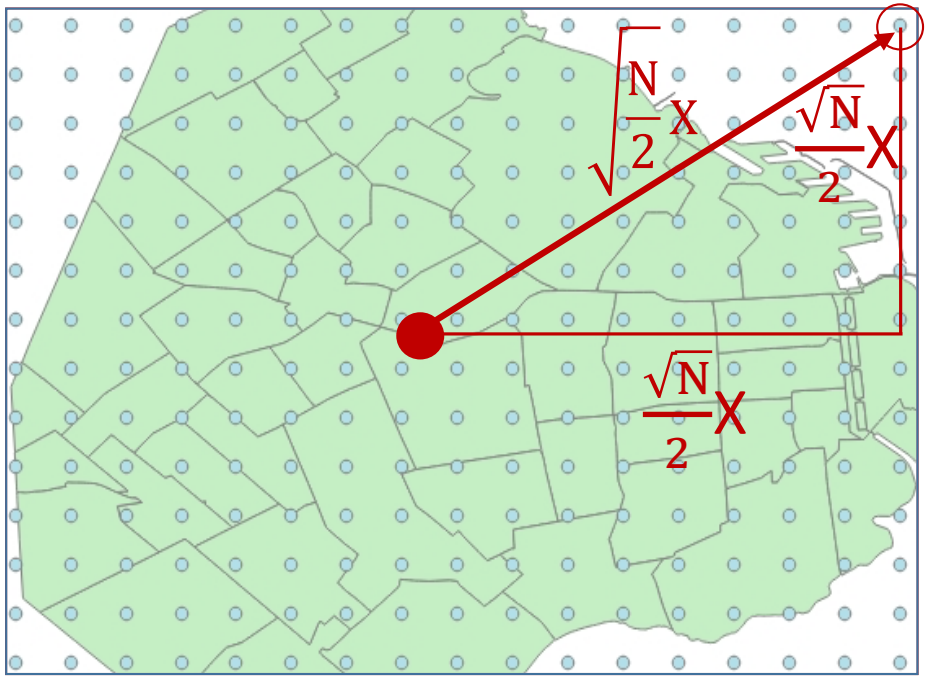
t = T = **0.01 s**

B) Center with N computers: $t = \frac{T}{\sqrt{N}} + 2\sqrt{\frac{N}{2}}\,X$

$= \frac{0.01}{10} + 2\sqrt{\frac{100}{2}}\,0.0001$

= 0.001 + 0.00142 = **0.00242 s**

Centralized scheme is 4x faster



Reasoning about the computing continuum
(c) With response time bound, B

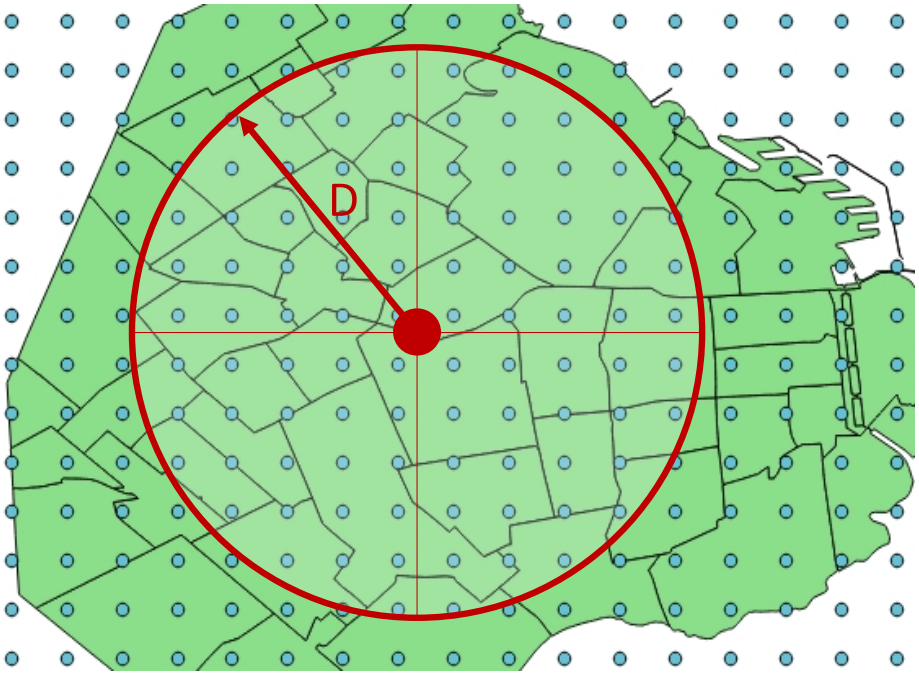
We want to know D for which:

$$\frac{T}{\sqrt{size}} + 2D \leq B$$

As *size* is $\pi D^2/X^2$, we want to solve:

$$\frac{T}{\sqrt{\pi D^2/X^2}} + 2D = B$$

From **A1**, there are $\pi D^2/X^2$ consumers within distance D of a compute center



Reasoning about the computing continuum
(c) With response time bound, B

We want to know D for which:

$$\frac{T}{\sqrt{size}} + 2D \leq B$$

As *size* is $\pi D^2/X^2$, we want to solve:

$$\frac{T}{\sqrt{\pi D^2/X^2}} + 2D = B$$

With B=0.01, T=0.001, X=0.0001 sec:

D = 0.004964 sec (~1000 km)

Then:

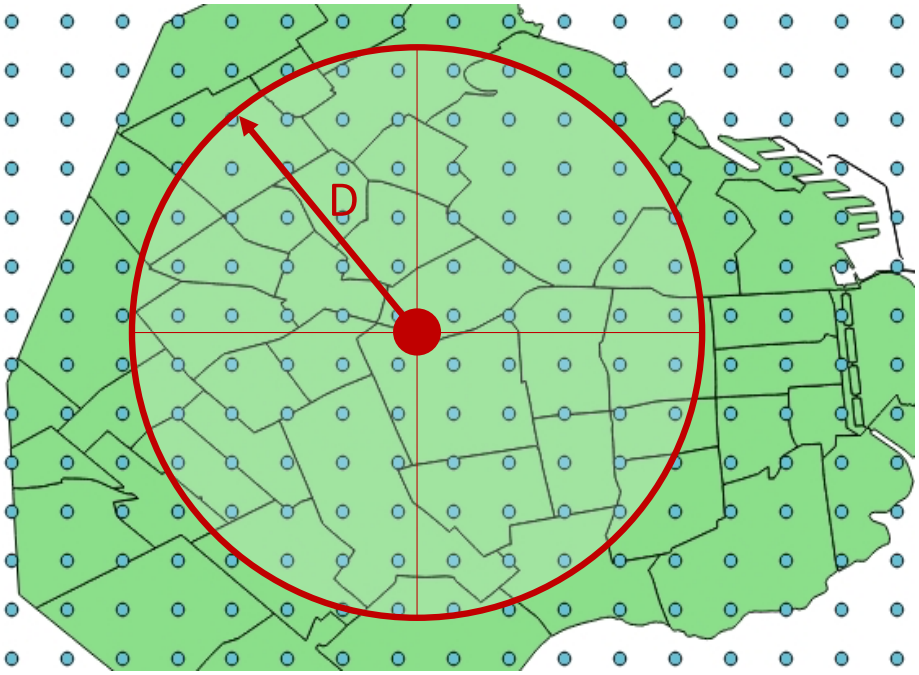
Size = $\pi D^2/X^2 = 7854$

Max processing time is

$2 \times 0.004964 + 0.001/\sqrt{7854}$

= 0.01 seconds

From **A1**, there are $\pi D^2/X^2$ consumers within distance D of a compute center



Reasoning about the computing continuum
(d) Discussion

The model emphasizes the importance of **distribution** ("disintegration") of function and **aggregation** of capability

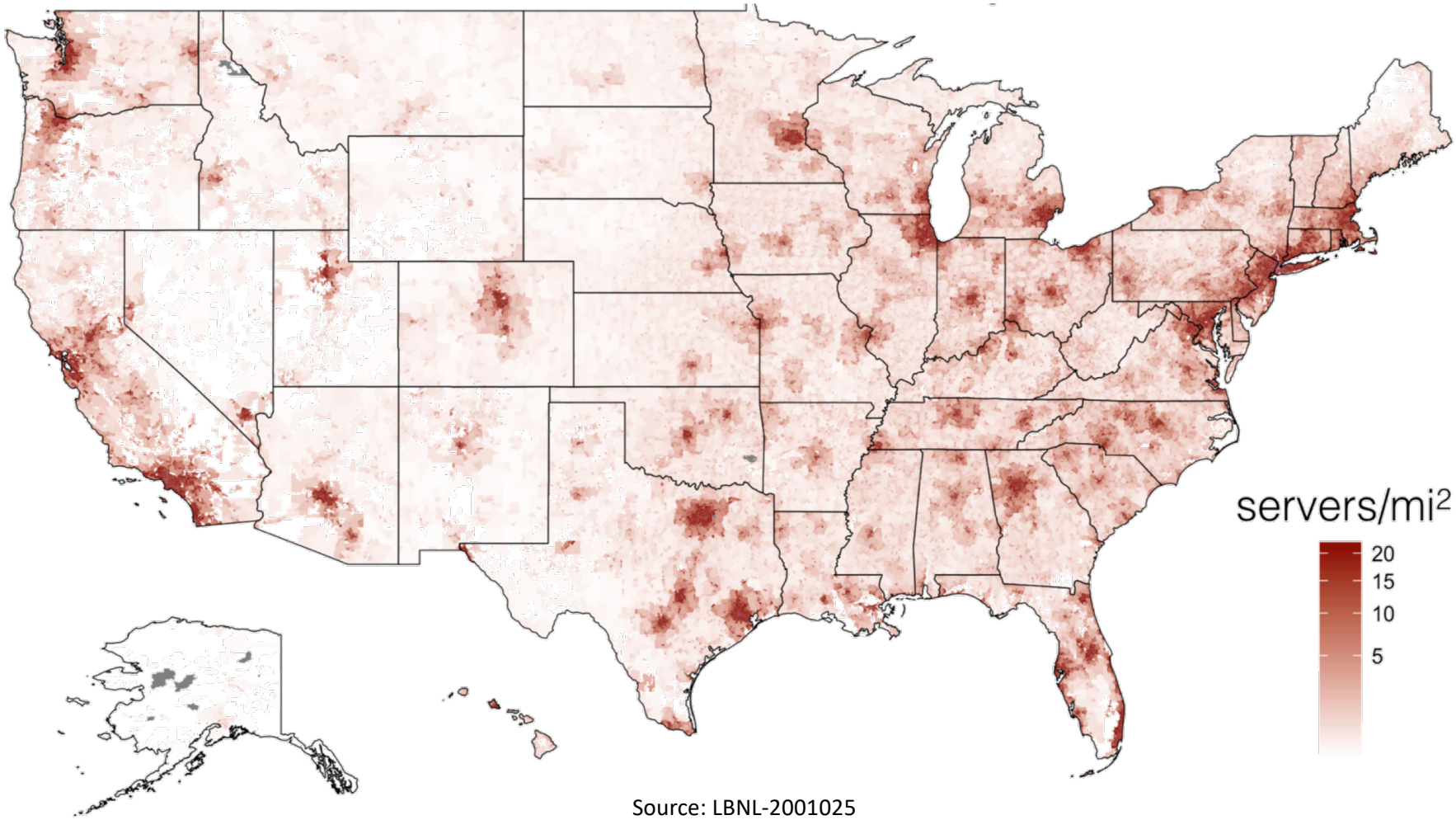
The model can be improved:

- Empirical data on scaling of cost and speed with size
- Data transfer costs
- Empirical data on workloads

Optimal solutions will involve compute centers of multiple sizes

→ Not just "center" and "edge"

Small and midsize data centers: Server intensity



Source: LBNL-2001025

Coding the continuum

Code: verb.

1) to arrange or enter in a code

Coding the continuum

Code: verb.

1) to arrange or enter in a code

2) to write code for

Coding the continuum

Code: verb.

1) to arrange or enter in a code

2) to write code for

Now that the machine has disintegrated across the net, how do we program it?

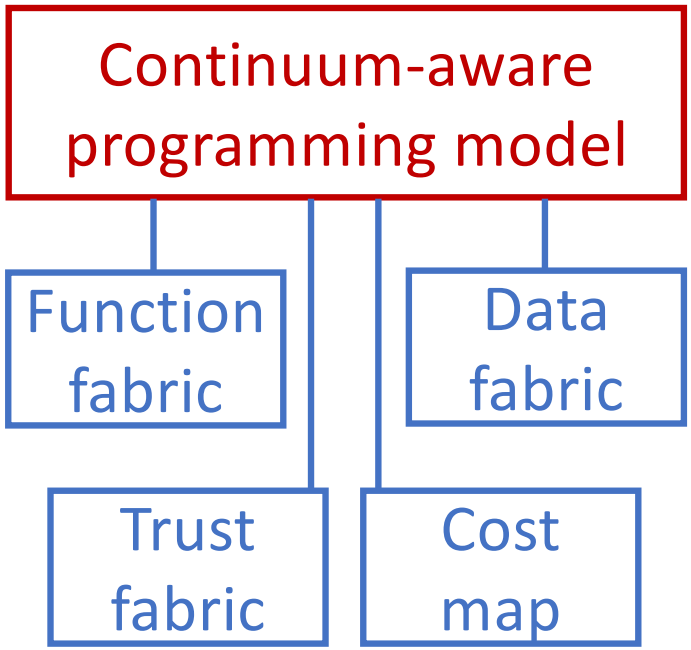
Coding the continuum

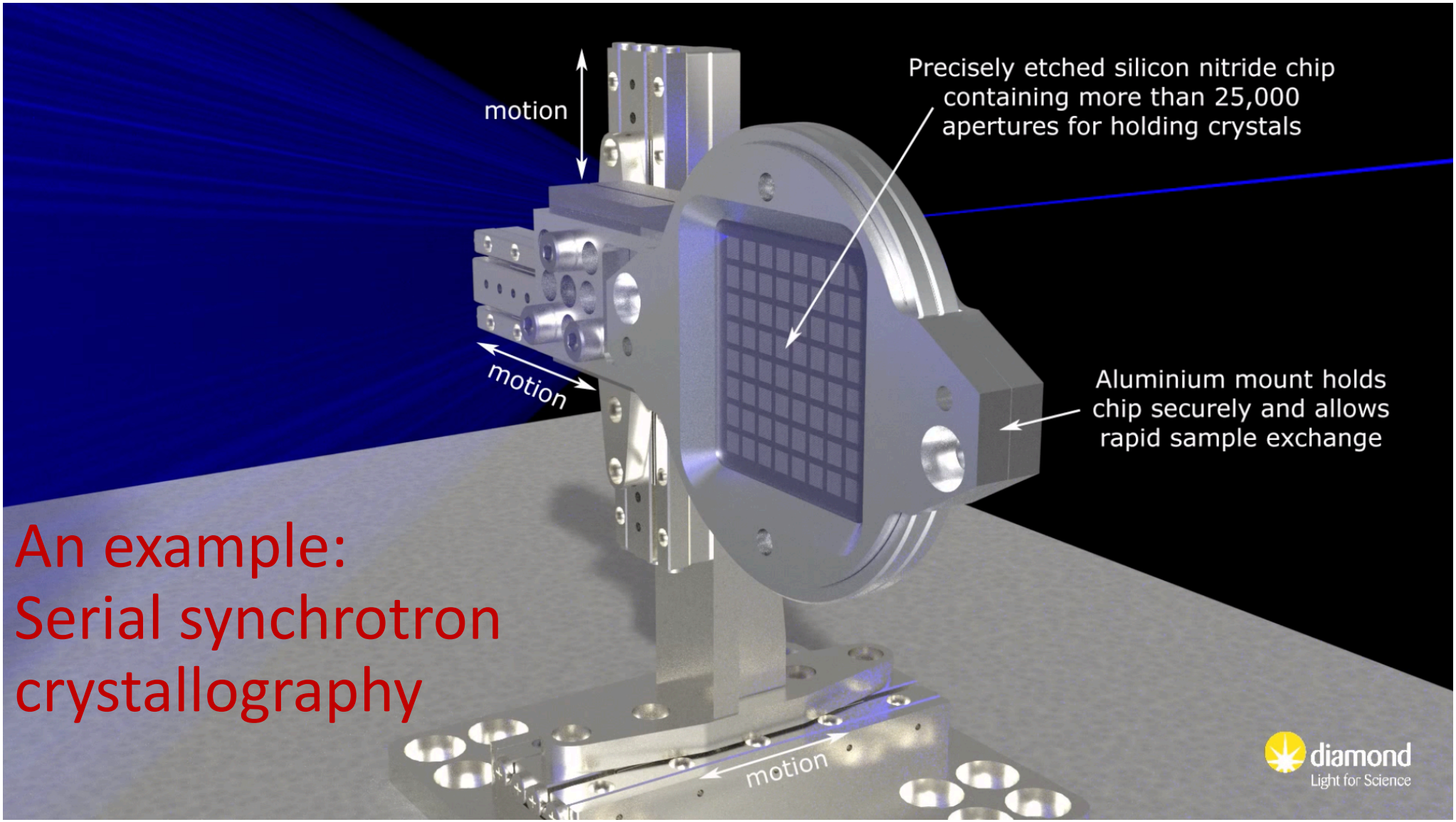
Code: verb.

1) to arrange or enter in a code

2) to write code for

Now that the machine has disintegrated across the net, how do we program it?



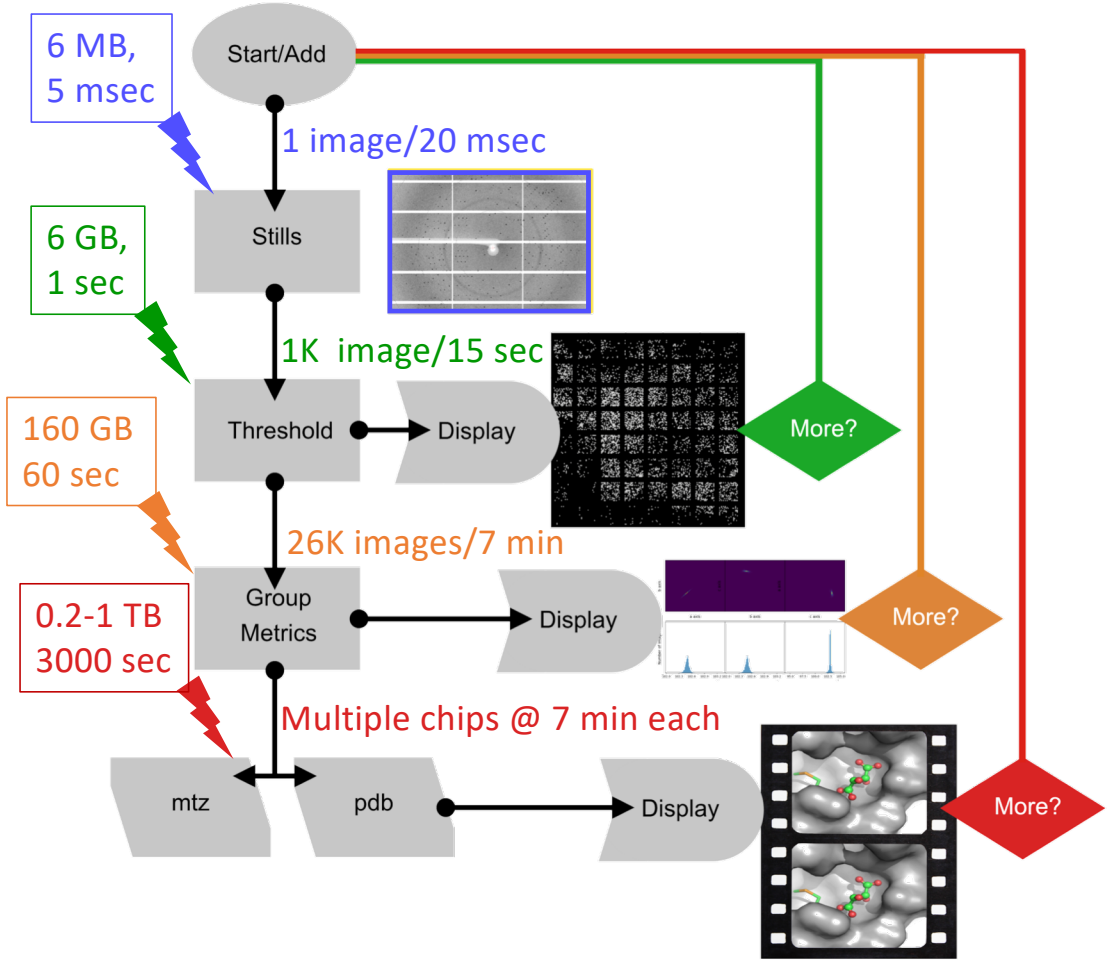


An example: Serial synchrotron crystallography

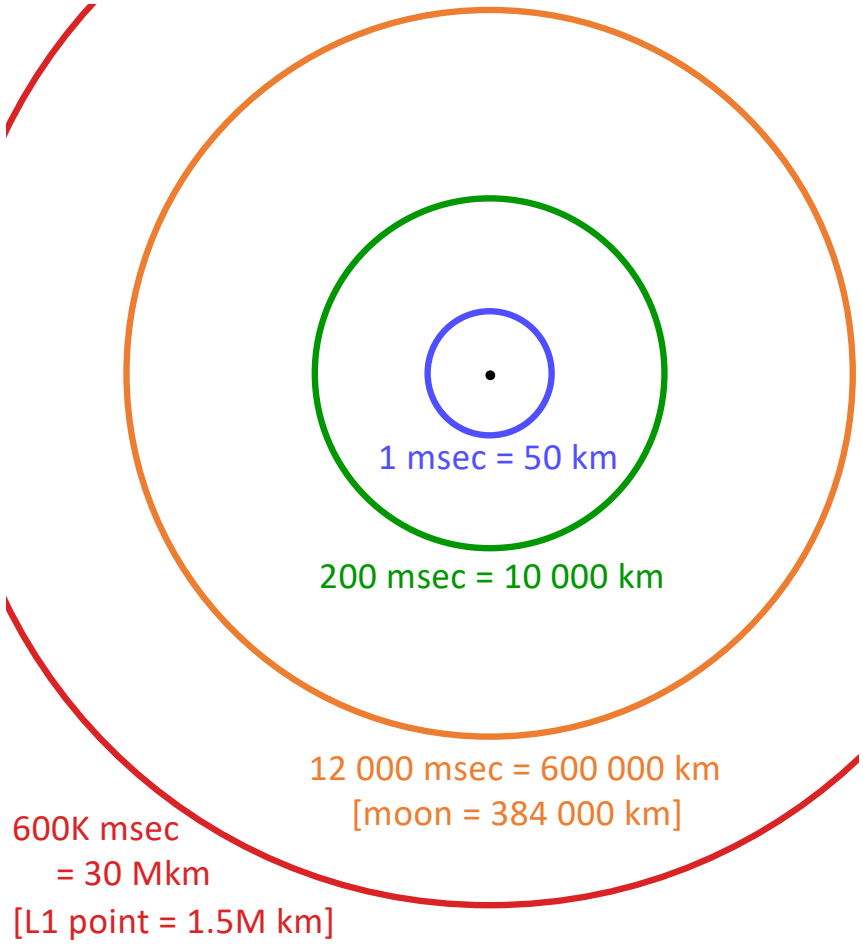
An example: Serial synchrotron crystallography

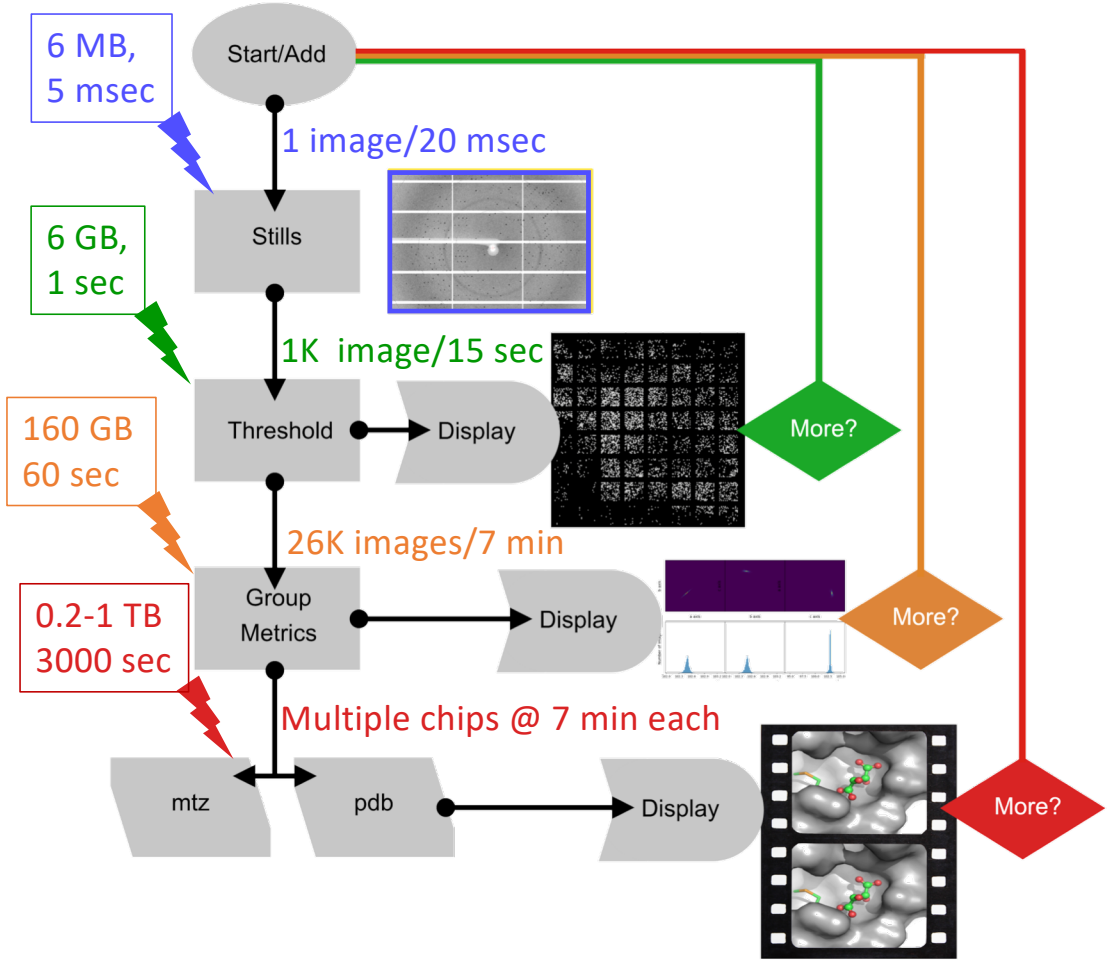
For each sample:

- Image crystals at ~50 Hz:
  - Validate each image
  - After 1000, quality control
  - After 26000, full analysis
- If good:
  - Determine crystal structure
  - Return crystal structure



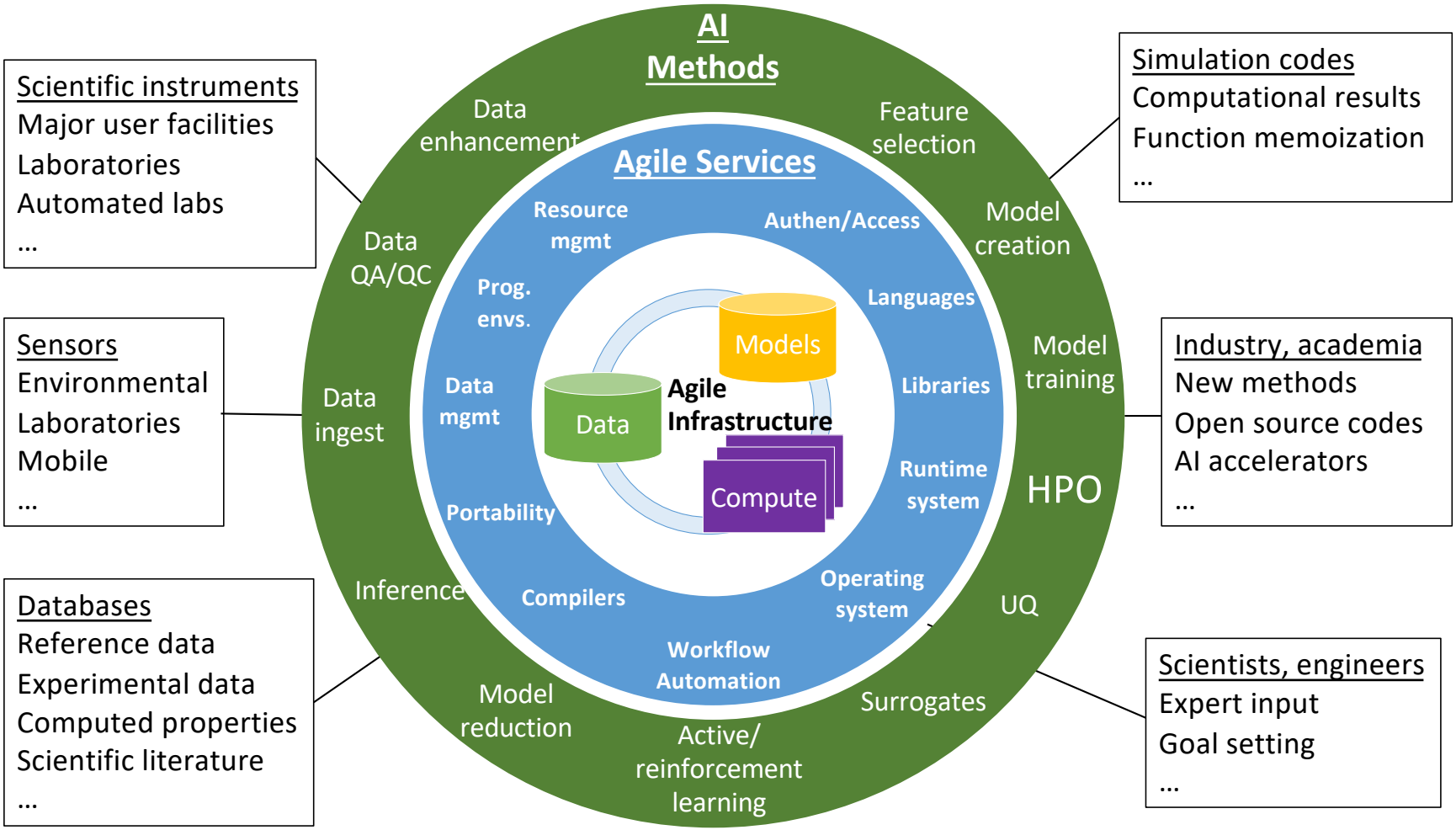
Coding the continuum: Serial crystallography



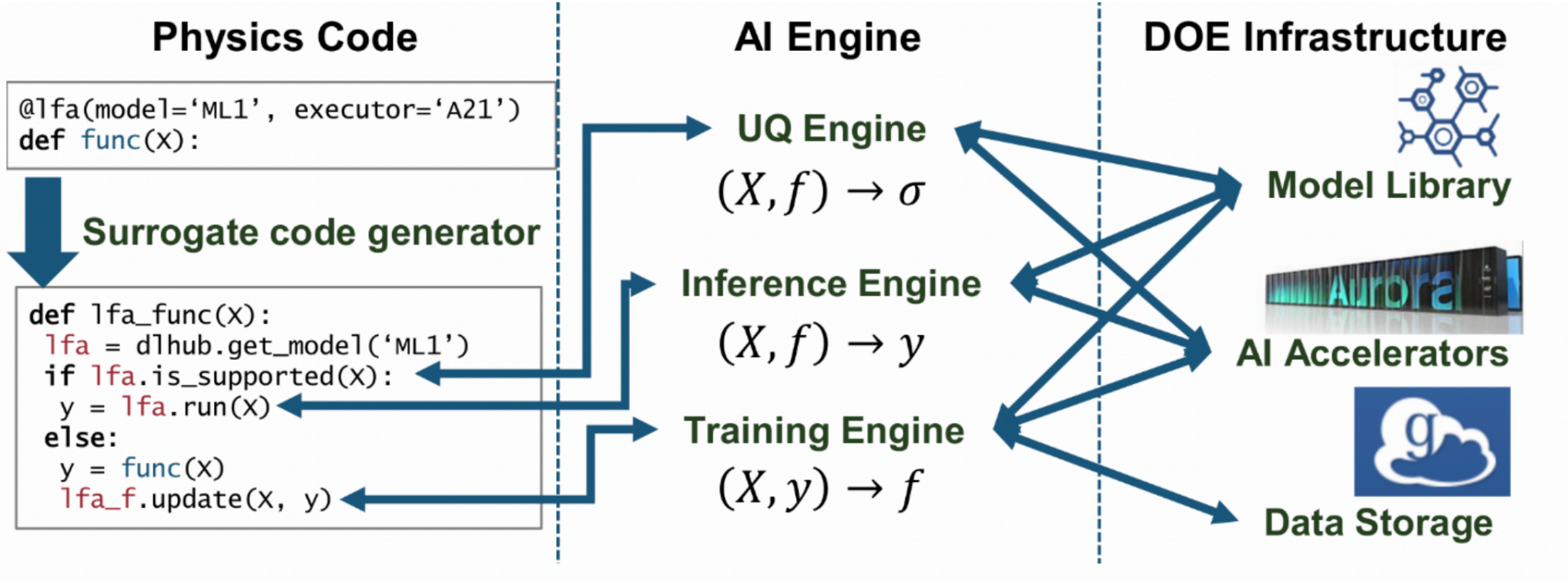



Advanced Photon Source
1 km
10 µsec
RTT
Argonne Leadership Computng Facility

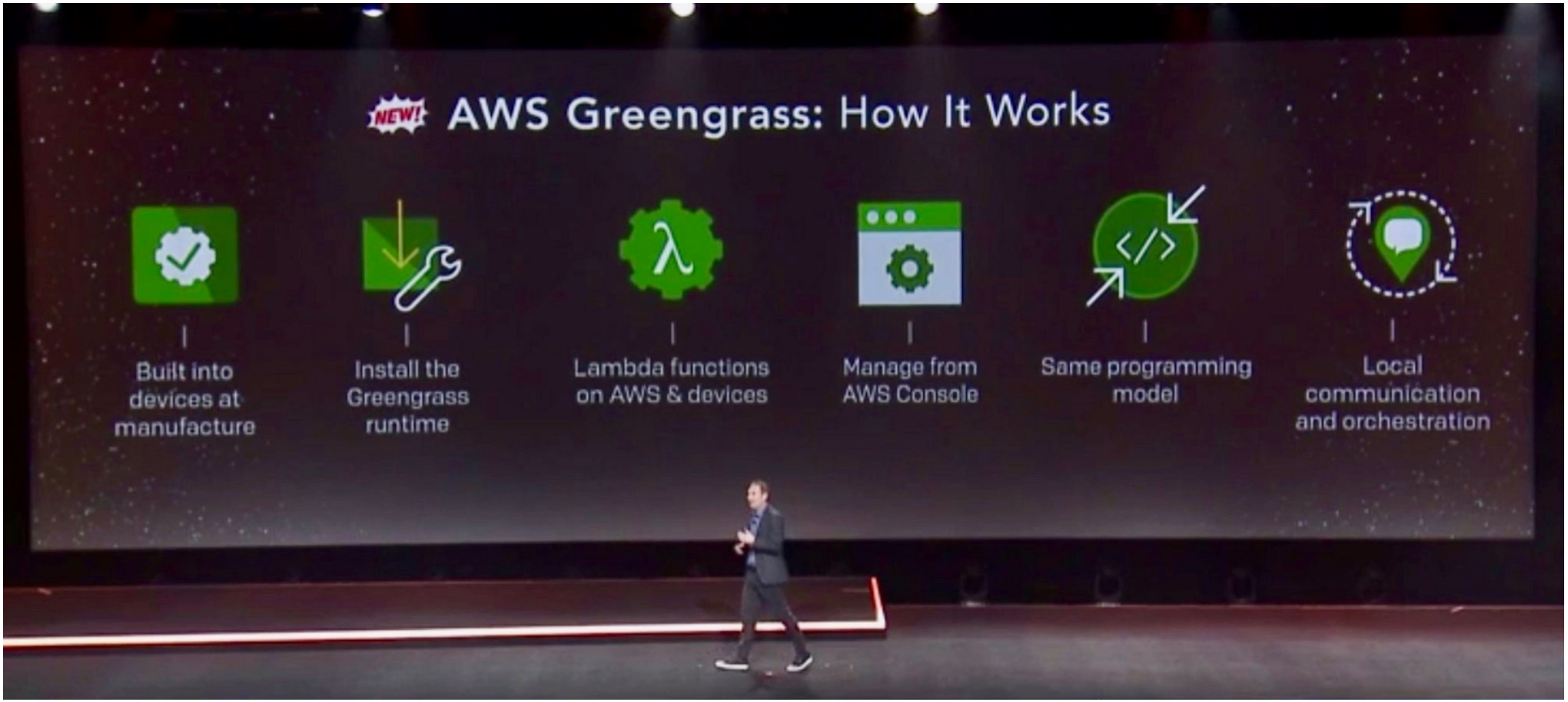
Similar needs arise across modern (AI-enabled) science



Learned Function Accelerators (LFAs)



Coding the continuum: Closed solution



https://read.acloud.guru/aws-greengrass-the-missing-manual-2ac8df2fbdf4

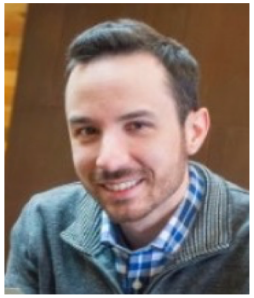
Coding the continuum: Elements of an open solution

Thanks to colleagues, especially:

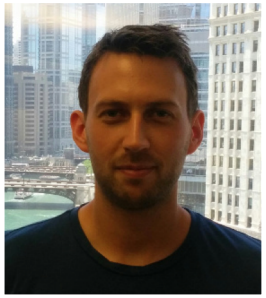
globus labs


Rachana Ananthakrishnan


Yadu Babuji


Ben Blaiszik


Kyle Chard


Ryan Chard


Zhuozhao Li


Tyler Skluzacek


Steve Tuecke

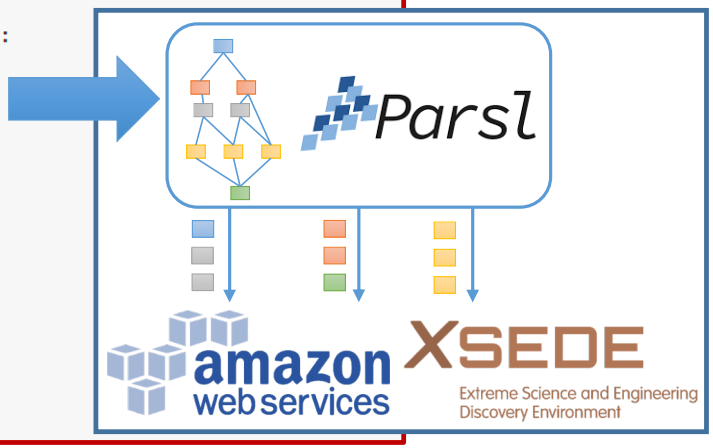

Anna Woodard


Logan Ward

Model registry	DLHub
Flows	Automate
Write programs	Parsl
Cost map	SCRIMP
Function fabric	funcX
Data fabric	Data services
Trust fabric	Auth

Coding the continuum: Elements of an open solution

```python
# App that estimates pi by placing points in a box
@python_app
def pi(total):
    import random

    # Set the size of the box (edge length) in which we drop random points
    edge_length = 10000
    center = edge_length / 2
    c2  = center ** 2
    count = 0

    for i in range(total):
        # Drop a random point in the box.
        x,y = random.randint(1, edge_length),random.randint(1, edge_length)
        # Count points within the circle
        if (x-center)**2 + (y-center)**2 < c2:
            count += 1

    return (count*4/total)

# App that computes the average of the values
@python_app
def avg_points(a, b, c):
    return (a + b + c)/3

# Estimate three values for pi
a, b, c = pi(10**6), pi(10**6), pi(10**6)

# Compute the average of the three estimates
avg_pi  = avg_points(a, b, c)
```



https://arxiv.org/pdf/1905.02158

http://parsl-project.org

Model registry — DLHub

Flows — Automate

Write programs — Parsl

Cost map — SCRIMP

Function fabric — funcX

Data fabric — Data services

Trust fabric — Auth

Coding the continuum: Elements of an open solution

```
In [1]: from funcx_sdk.client import FuncXClient

        fxc = FuncXClient()

In [2]: func = """
        def add(data):
            sum_val = sum(data['data'])
            return sum_val
        """

In [3]: fxc.register_function("add_func", func, description="Sum a list of numbers.")

In [4]: input_data = [1, 2, 3]

        res = fxc.run(input_data, "user#laptop", "add_func")

In [5]: print(res)
        6
```

Portable code

Python
Docker, Shifter, Singularity

Any access

SSH, Globus, cluster or HPC scheduler

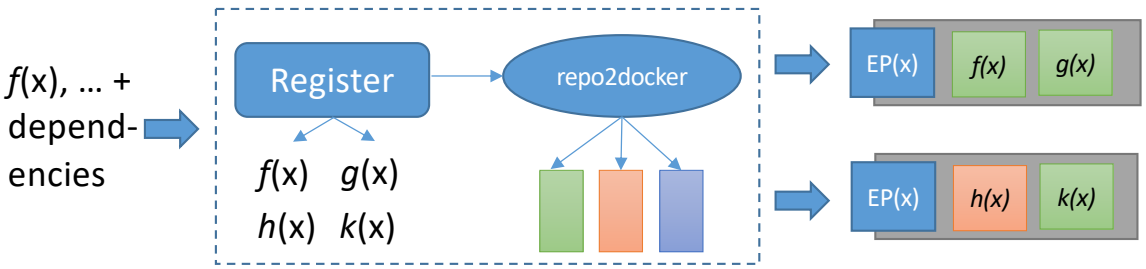
Any computer

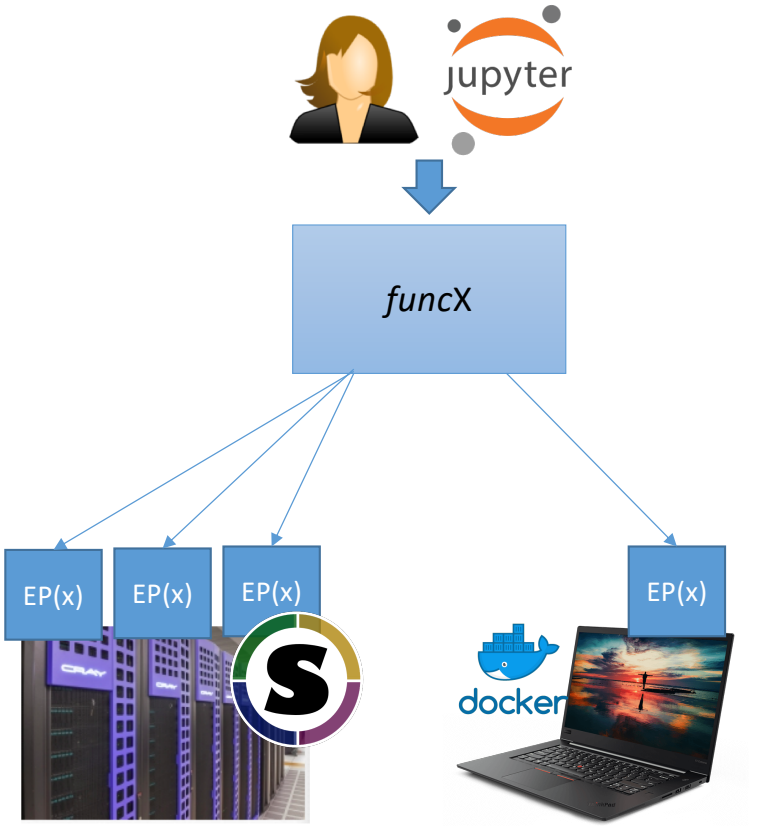
Clusters, clouds, HPC, accelerators

Model registry	DLHub
Flows	Automate
Write programs	Parsl
Cost map	SCRIMP
Function fabric	funcX
Data fabric	Data services
Trust fabric	Auth

funcX: Transform clouds, clusters, and supercomputers into high-performance function serving systems

```
In [1]: from funcx_sdk.client import FuncXClient

        fxc = FuncXClient()

In [2]: func = """
        def add(data):
            sum_val = sum(data['data'])
            return sum_val
        """

In [3]: fxc.register_function("add_func", func, description="Sum a list of numbers.")

In [4]: input_data = [1, 2, 3]

        res = fxc.run(input_data, "user#laptop", "add_func")

In [5]: print(res)
        6
```

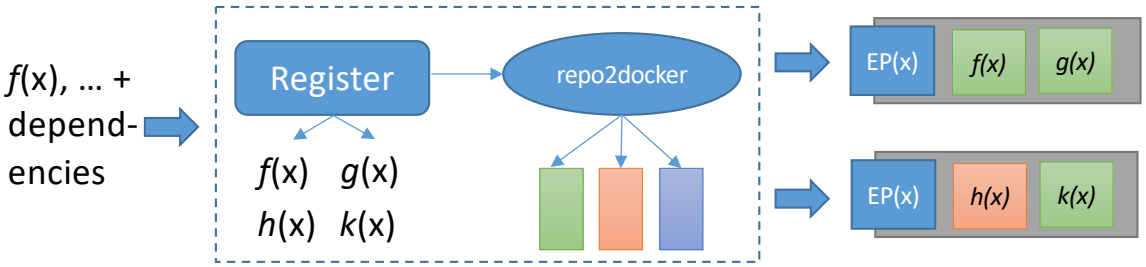


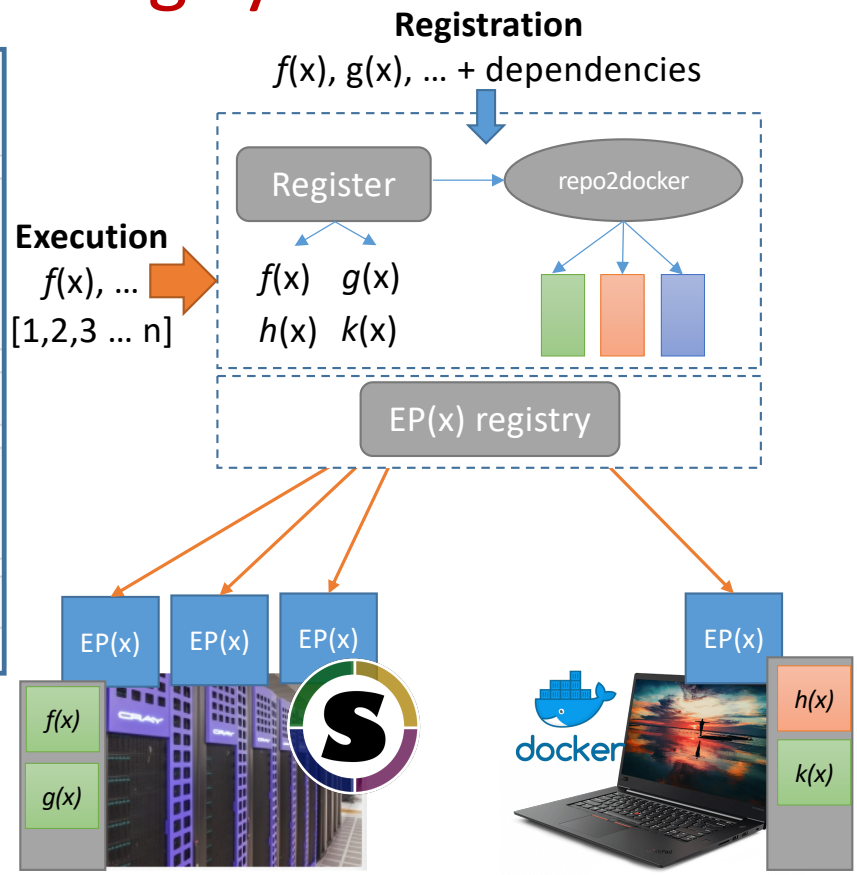


Simply deploy *funcX* endpoint to transform a computer into a function serving system

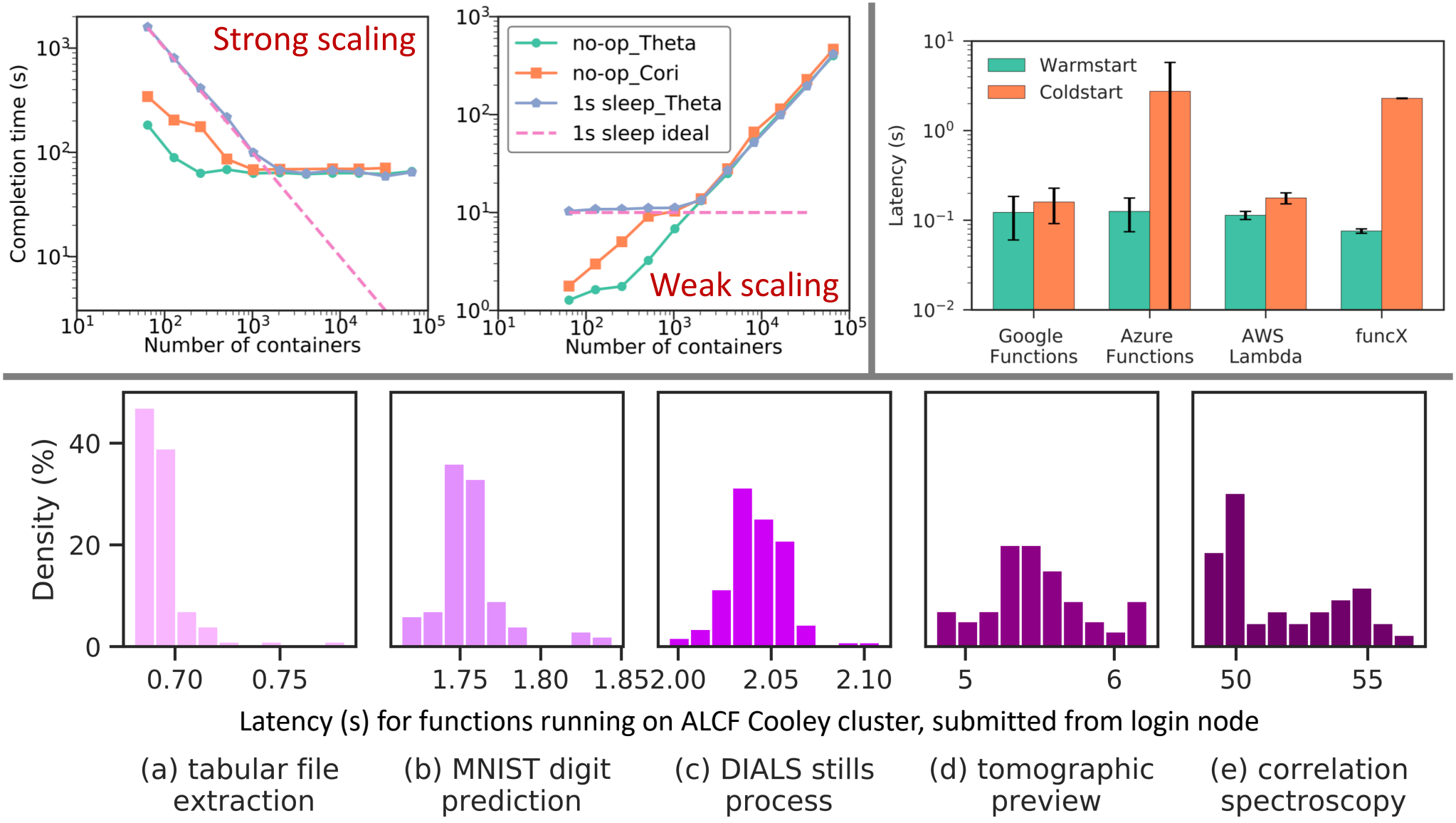
funcX: Transform clouds, clusters, and supercomputers into high-performance function serving systems

```
In [1]: from funcx_sdk.client import FuncXClient

        fxc = FuncXClient()

In [2]: func = """
        def add(data):
            sum_val = sum(data['data'])
            return sum_val
        """

In [3]: fxc.register_function("add_func", func, description="Sum a list of numbers.")

In [4]: input_data = [1, 2, 3]

        res = fxc.run(input_data, "user#laptop", "add_func")

In [5]: print(res)
        6
```





Simply deploy *funcX* endpoint to transform a computer into a function serving system


Strong scaling
Completion time (s)
Number of containers
Weak scaling
no-op_Theta
no-op_Cori
1s sleep_Theta
1s sleep ideal
Number of containers
Warmstart
Coldstart
Latency (s)
Google Functions
Azure Functions
AWS Lambda
funcX
Density (%)
Latency (s) for functions running on ALCF Cooley cluster, submitted from login node
(a) tabular file extraction
(b) MNIST digit prediction
(c) DIALS stills process
(d) tomographic preview
(e) correlation spectroscopy

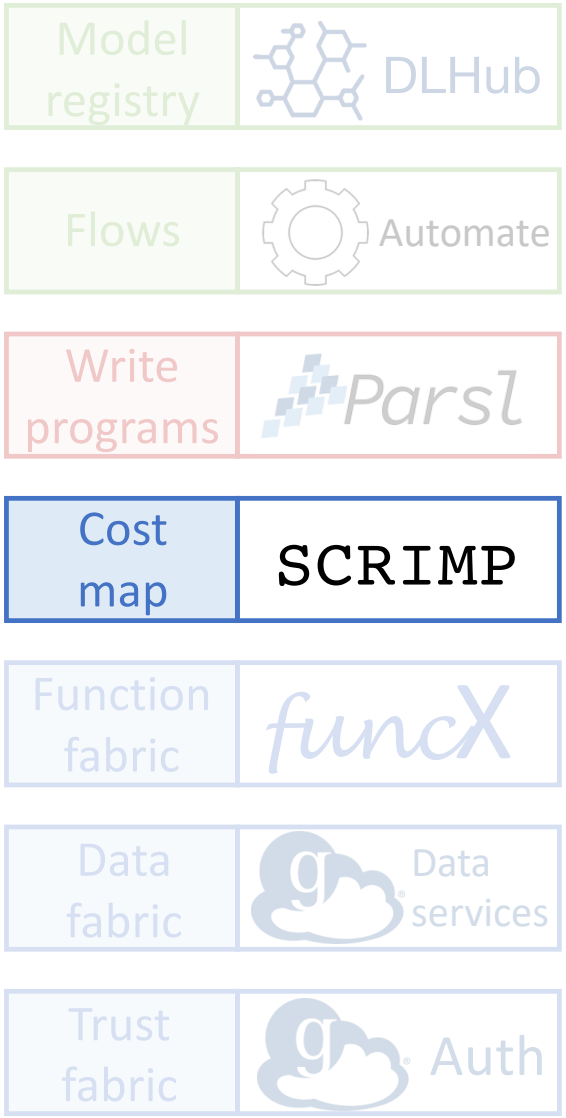
Common FaaS systems, compared

	Function Language	Intended Infrastructure	Virtualization	Triggers	Maximum Walltime (s)	Billing
Amazon Lambda	C#, Go, Java, Powershell, Ruby Python, Node.js	Public cloud, Edge (Greengrass)	Firecracker (KVM)	HTTP, AWS services	900	Requests, runtime, memory
Google Cloud Functions	BASH, Go, Node.js, Python	Public cloud	Undefined	HTTP, Pub/Sub, storage	540	Requests, runtime, memory
Azure Functions	BASH, Java, Python, Visual Studio	Public cloud, local	OS images	HTTP, APIM, MS services	600	Requests, runtime, SLA
OpenWhisk	Ballerina, Go, Java, Node.js, Python, Go	Kubernetes, Private cloud, Public cloud	Docker	HTTP, IBM Cloud OW-CLI	300	IBM Cloud: Requests, runtime Local: NA
Kubeless	Node.js, Python .NET, Ruby Ballerina, PHP	Kubernetes	Docker	HTTP, scheduled, Pub/Sub	Undefined	NA
SAND	C, Go, Java, Node.js, Python	Public cloud, Private cloud	Docker	HTTP, Internal event	Undefined	Triggers
Fn	Go, Java, Ruby, Node.js, Python	Public cloud, Kubernetes	Docker	HTTP, direct trigger	300	NA
Abaco	Container	TACC clusters	Docker	HTTP	Undefined	Undefined
funcX	Python	HPC, Public cloud	Singularity, Shifter, Docker	HTTP, Globus Automate	No limit	HPC SUs

Coding the continuum: Elements of an open solution

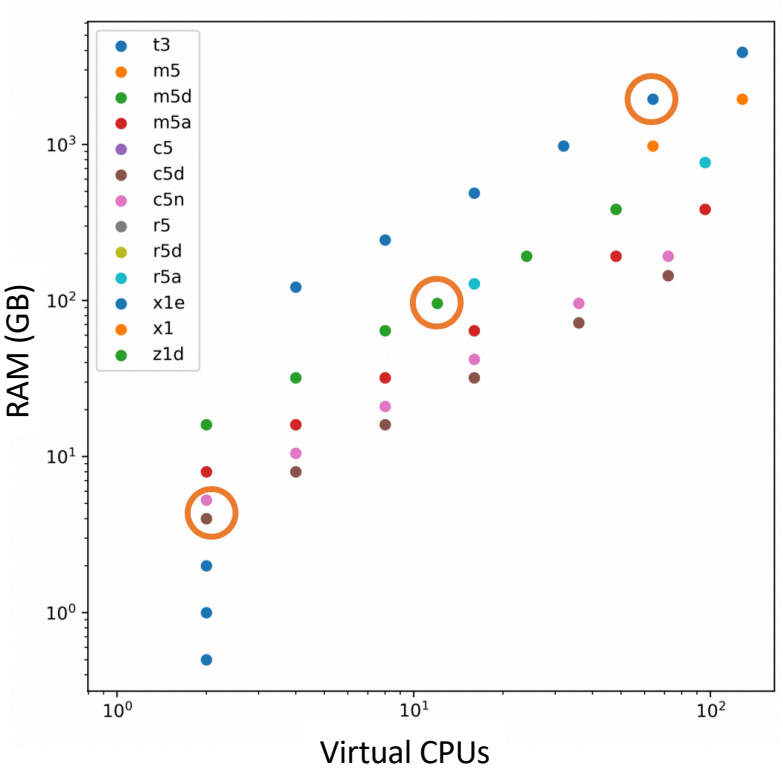
Incremental construction of a **personalized cost map**

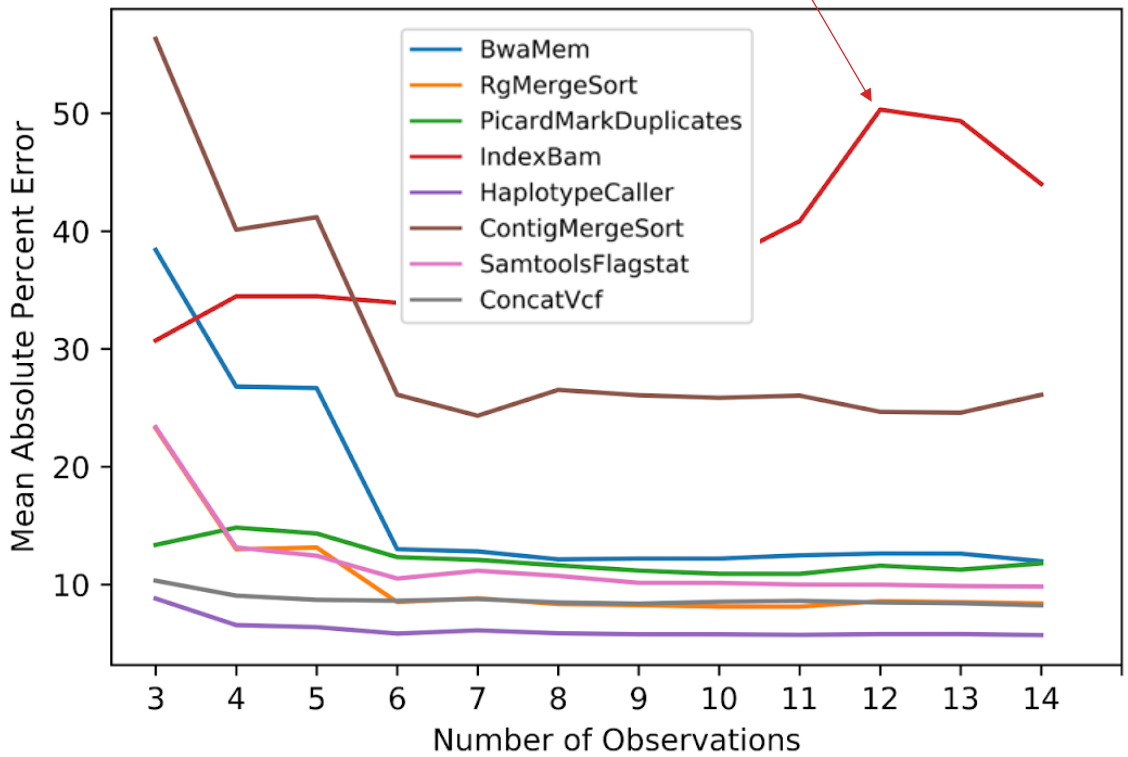
- Build black-box performance models from observed execution times for different codes on different platforms
- Transfer learning across codes, problem sizes, and hardware platforms
- Experiment design to choose experiments that maximize reduction in uncertainty
- Evolve models over time as codes and platforms change
- Use models for instance selection and scheduling



Example: A cost map for bioinformatics applications on different AWS instance types

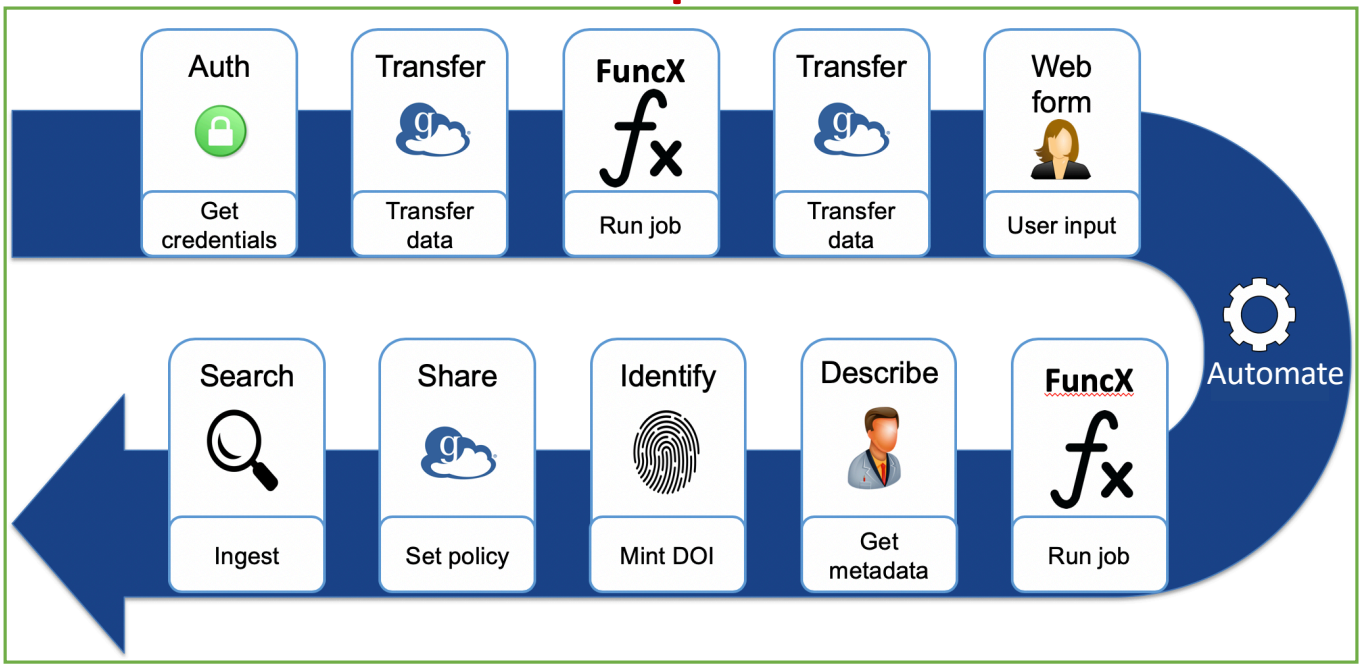
IndexBam performs better on compute-optimized instances. Poorly chosen experiments mislead the model





On average, within 30% of final error after 4 experiments and within 2.3% after 6

Coding the continuum: Elements of an open solution



Detect and respond to events

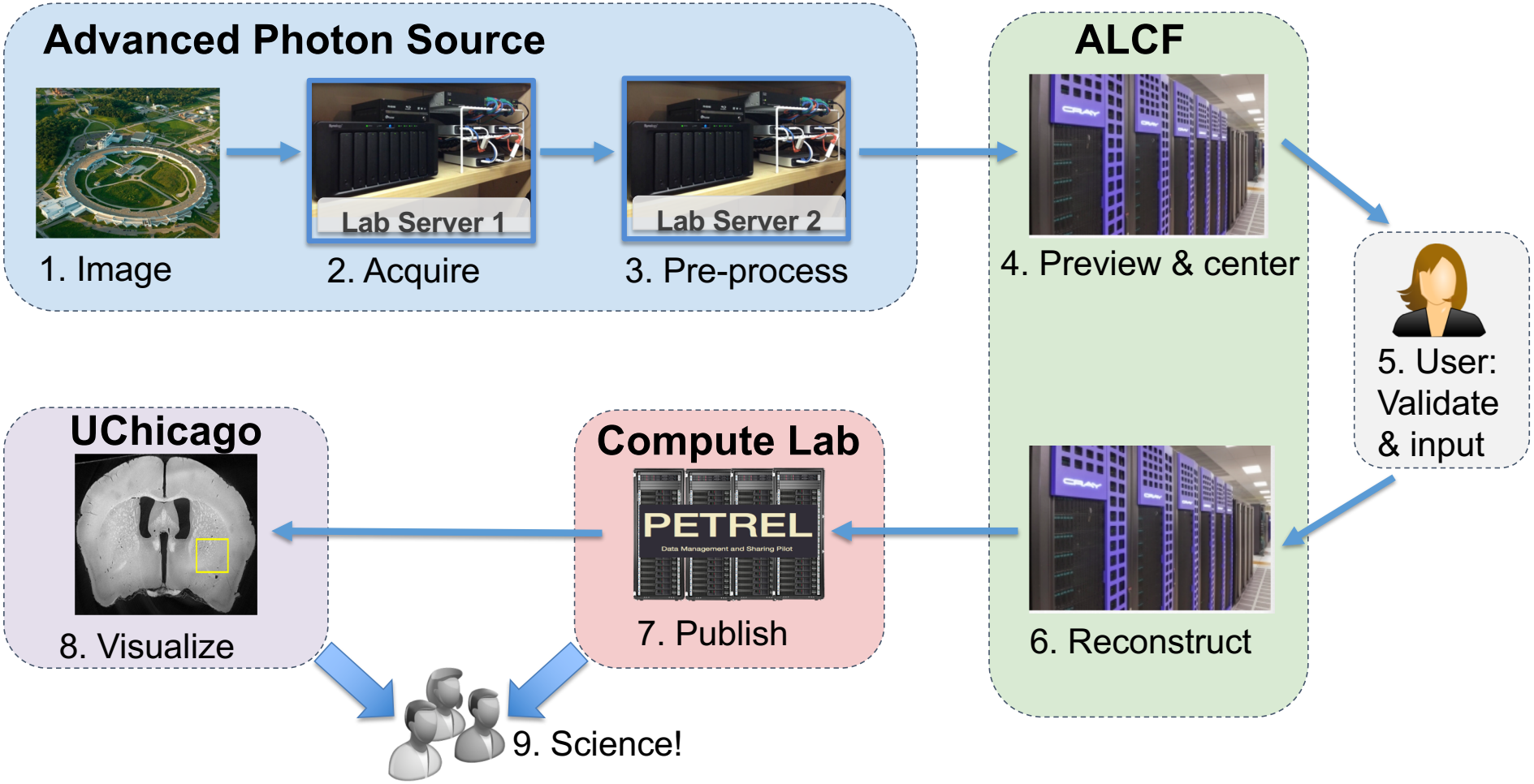
→ E.g., in HPC file systems: FSMon (Arnab Paul et al.)

Invoke RESTful services, and accept user input

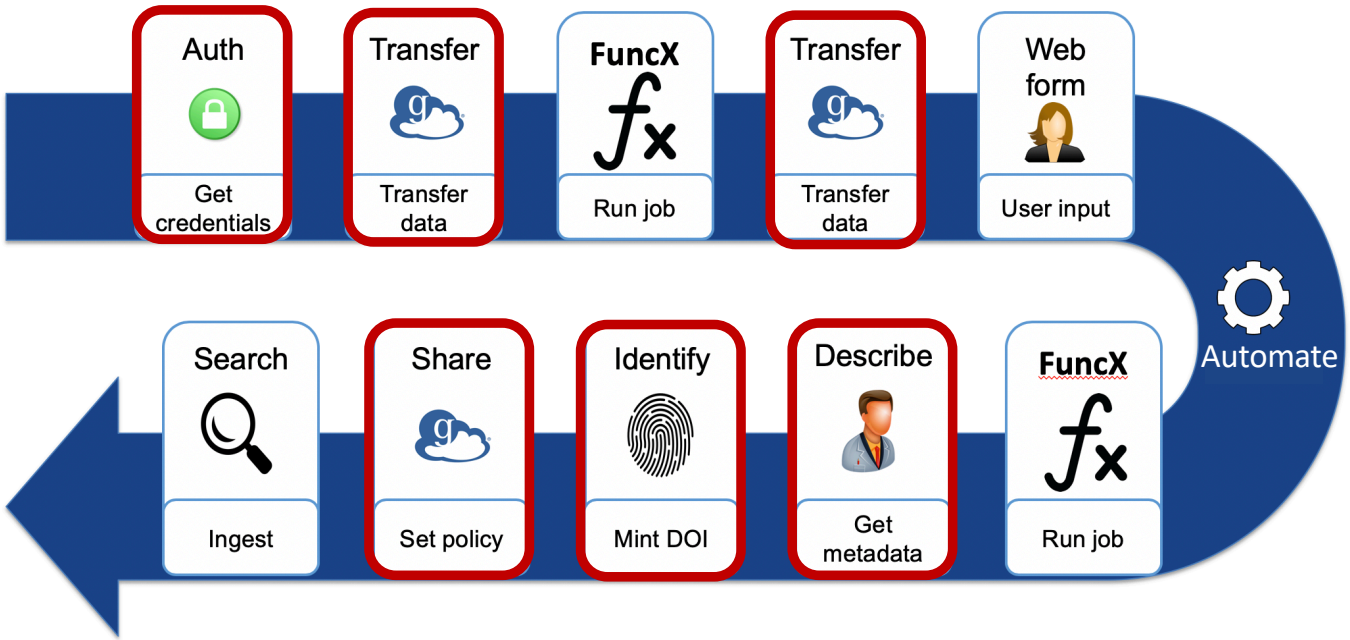
Manage short- and long-lived activities

Model registry	DLHub
Flows	Automate
Write programs	Parsl
Cost map	SCRIMP
Function fabric	funcX
Data fabric	Data services
Trust fabric	Auth

Flow automation in a neuroanatomy automation



Coding the continuum: Elements of an open solution



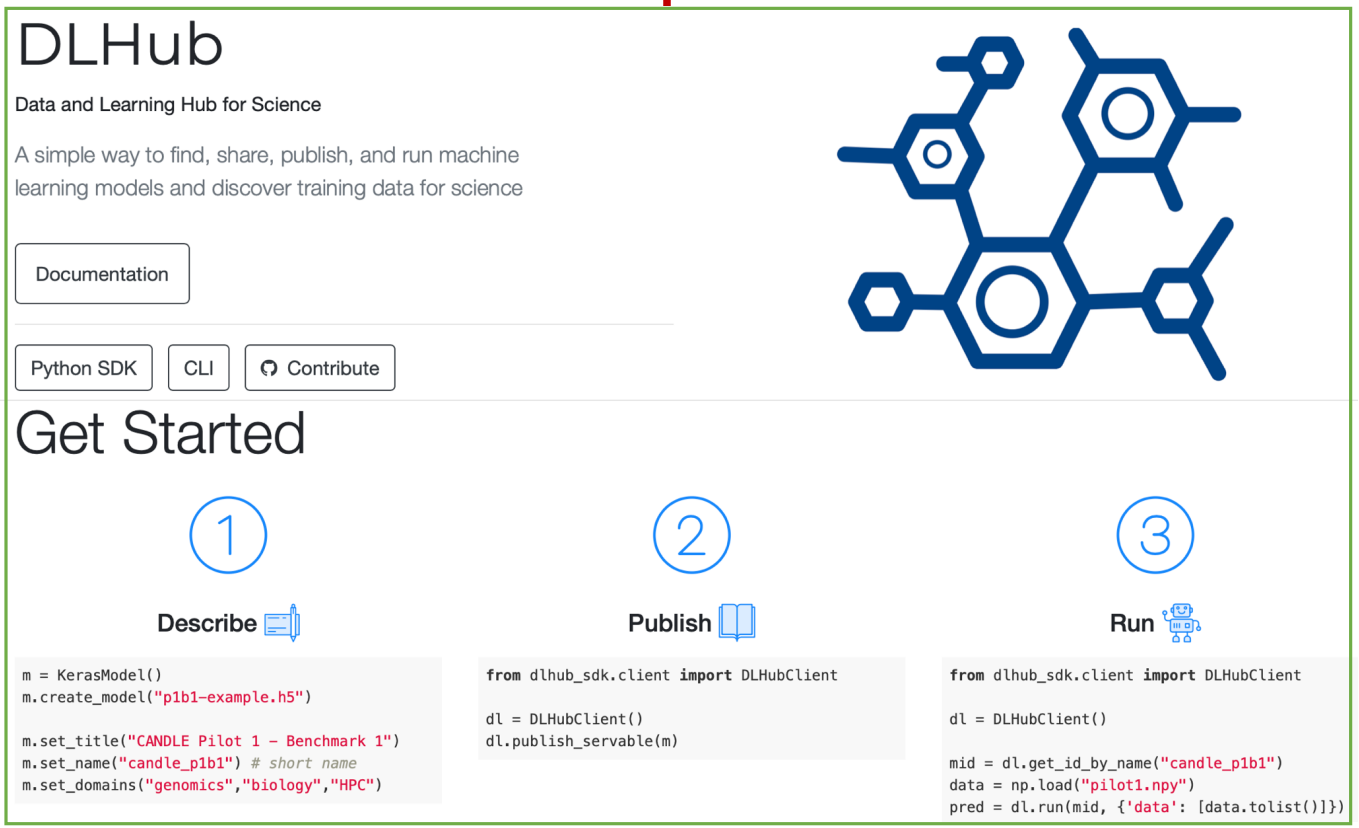
Cloud-hosted services support data lifecycle events

→ Cloud for high-reliability, modest-latency actions

→ Integrated OAuth-based security with delegation

Model registry	DLHub
Flows	Automate
Write programs	Parsl
Cost map	SCRIMP
Function fabric	funcX
Data fabric	Data services
Trust fabric	Auth

Coding the continuum: Elements of an open solution



https://arxiv.org/abs/1811.11213

Paper @ Session 7, 1:30pm today

dlhub.org

Model registry	DLHub
Flows	Automate
Write programs	Parsl
Cost map	SCRIMP
Function fabric	funcX
Data fabric	Data services
Trust fabric	Auth

Coding the Continuum: Thanks for support

US Department of Energy

US National Science Foundation

US National Institutes of Health

US National Institute of Standards and Technology

Amazon Web Services

Globus subscribers

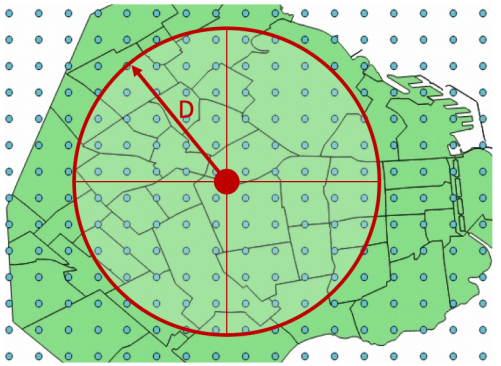
"the machine disintegrates across the net into a set of special-purpose appliances"

Coding the [location-speed] continuum

Code: verb:

1) to arrange or enter in a code

"Henceforth, **location** for itself, and **speed** for itself shall completely reduce to a mere shadow, and only some sort of union of the two shall preserve independence."



$$\frac{T}{\sqrt{\pi D^2/X^2}} + 2D = B$$

2) to write code for

Distribute computational tasks across a heterogeneous computing fabric

Cost map	SCRIMP	Data fabric	Data services	Write programs	Parsl	Model registry	DLHub
Function fabric	funcX	Trust fabric	Auth			Flows	Automate

foster@anl.gov

labs.globus.org – dlhub.org – globus.org – parsl-project.org