

# Lecture 1 - Introduction

Stanford CS343D (Fall 2021)  
Fred Kjolstad

# Course staff



Fred Kjolstad



Caleb Donovanick

# Administria

- Syllabus at <https://cs343d.github.io>
- Discussion will happen through Piazza:
  - Sign up at: [piazza.com/stanford/fall2021/cs343d](https://piazza.com/stanford/fall2021/cs343d)
- Office Hours (Zoom scheduled through Canvas)
  - Fred: Wednesday 10–11am
  - Caleb: Monday 5–6pm

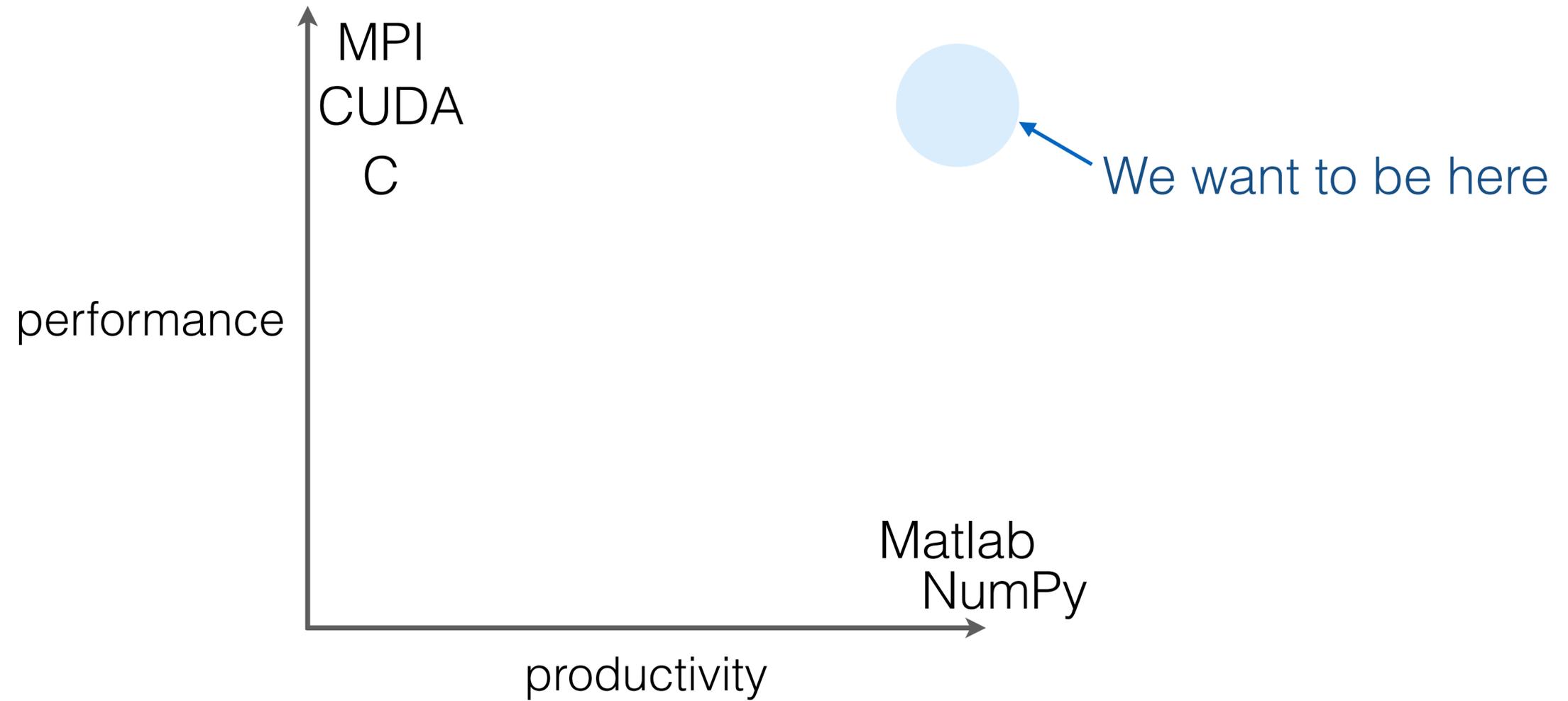
# Goals of the Course

- Introduce you to domain-specific and collection-oriented programming languages from the past
- Introduce you to compiler techniques to get good performance for dense and sparse applications
- Bring you to one of the frontiers of PL and compiler research
- Get you thinking about abstractions and semantics
- Abstraction, abstraction, abstraction

# Expectations

- Read papers and engage in class (25%)
  - Tuesdays and Thursdays 9:45 — 11:15
  - ~2 readings per class
  - Classes will have a lecture followed by paper discussion
  - Everyone will get a chance to lead a discussion
- Two assignments (20%)
  - MiniAPL
  - Relational query implementation
- Essay (15%)
- Project (40%)

# It is all about performance and productivity



# Performance translates to less time and less energy



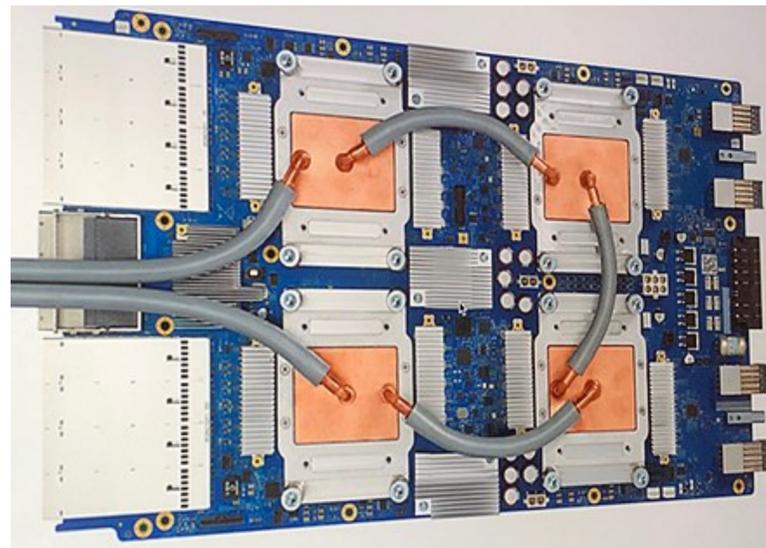
Data centers



Supercomputers



Self-driving cars



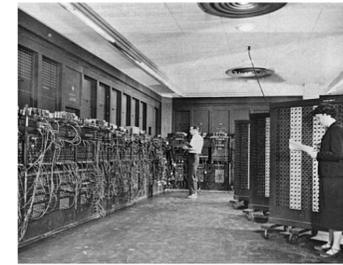
Tensor Processing Unit



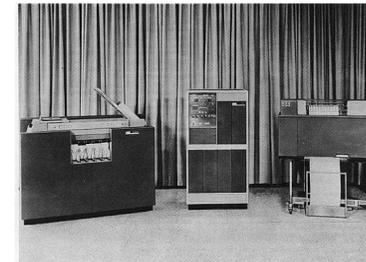
Cell-phone batteries

# Eras of Computing

Era of simulation (1945–1965)



Era of data processing (1965–1984)



Era of Personal Computing (1984–1995)



Era of communication (1995–2015)

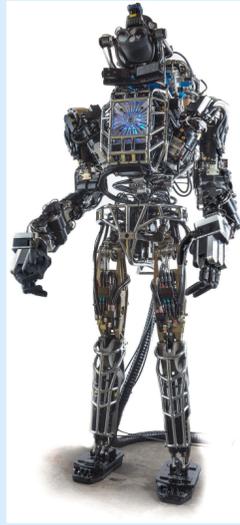


Era of interaction (2015–????)



# Modern applications are performance hungry

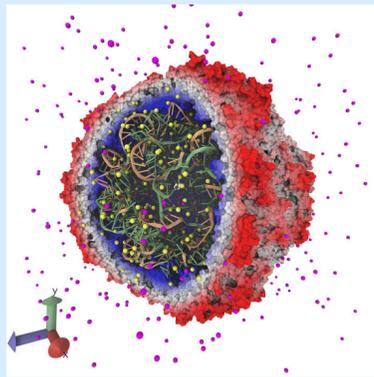
## Simulation and Optimization



Robotics



Graphics Simulations

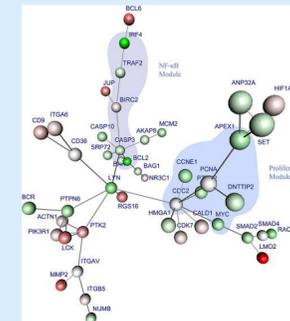


Virus Modelling

## Data Analytics



Social Networks



Computational Biology

Kristina  
★★★★★ Great Product  
March 30, 2017  
Color: White | Verified Purchase  
Great product. Large enough for all spoons and fits nicely on my stovetop. Would definitely buy it again.

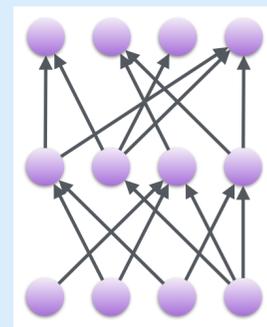
Teresa  
★★★★★ Excellent buy  
October 25, 2017  
Verified Purchase  
This is a great product for your boy who loves sports! It was a good value as well. Other stores sell for 3x the cost. I bought one for a basketball and football and my 9-year old loves it in his room. Solid item too, not flimsy. Will hold items nicely.

Lisa  
★★★★★ I was really disappointed. The spoon holder it self was great and ...  
December 31, 2016  
Color: Black | Verified Purchase  
This product came with a manufacture's chips in it. It is not the sellers fault but I do not know how many in this batch this seller may have. I was really disappointed. The spoon holder it self was great and larger then I expected.

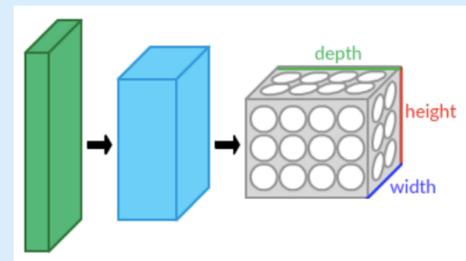
Sarah  
★★★★★ Malfunctioned within a month. Waste of \$.  
December 5, 2017  
Style: Battery Powered Alarm | Size: 1 Pack | Verified Purchase  
I chose this one because the reviews were good. It malfunctioned within a month. The back of the alarm has a key for the chips and of course mine was a lemon. It looks like it was just made August 9th, 2017. I received it at the end of October and it died mid-November. It was a waste of money.

Recommender Systems

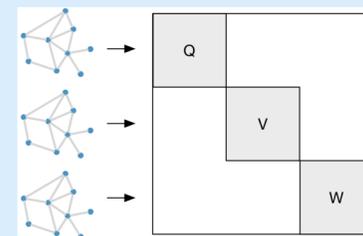
## Machine Learning



Sparse Networks

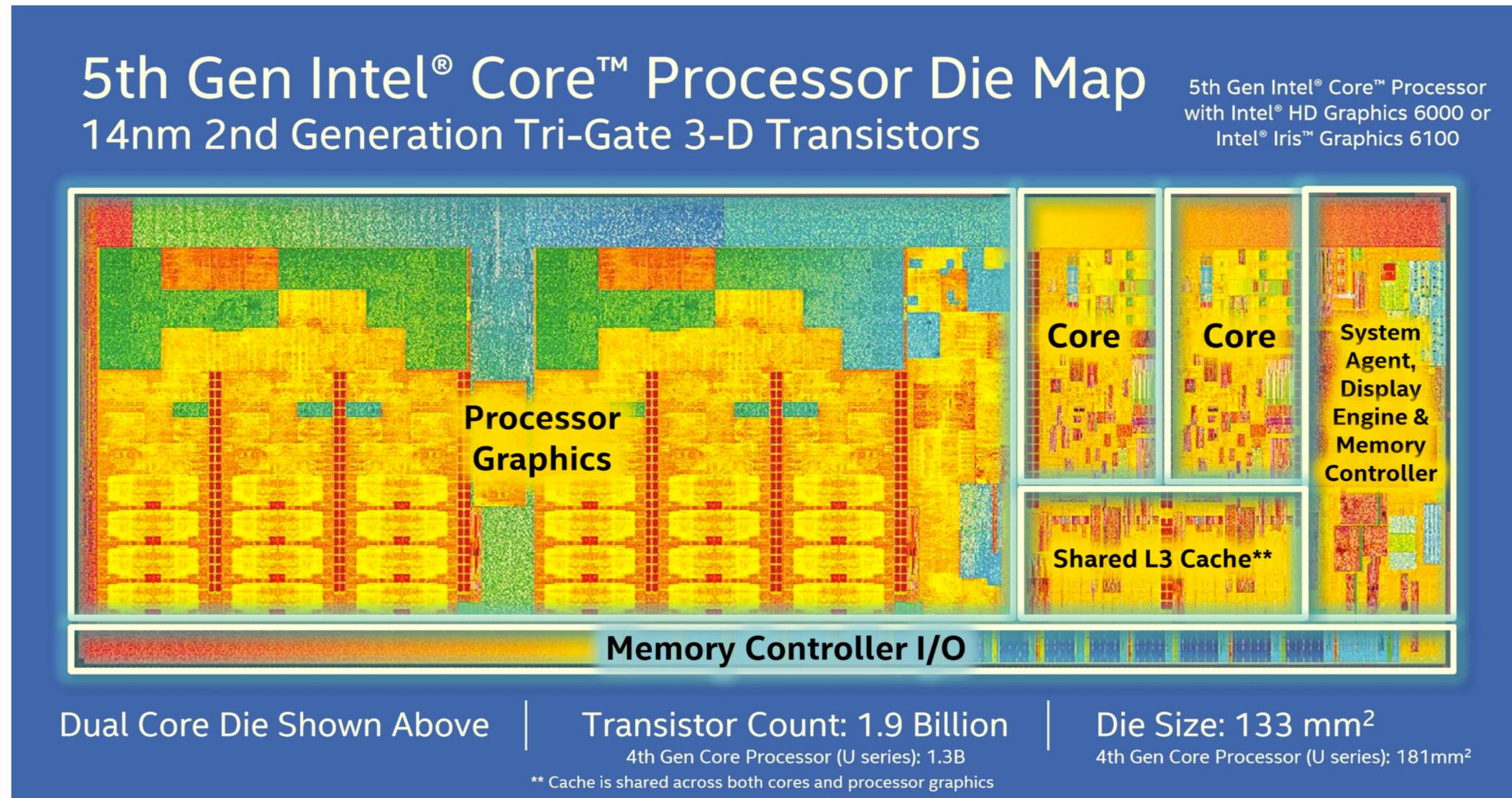


Sparse Convolutional Networks



Graph Convolutional Network

# Modern hardware is heterogeneous and programming it is hard



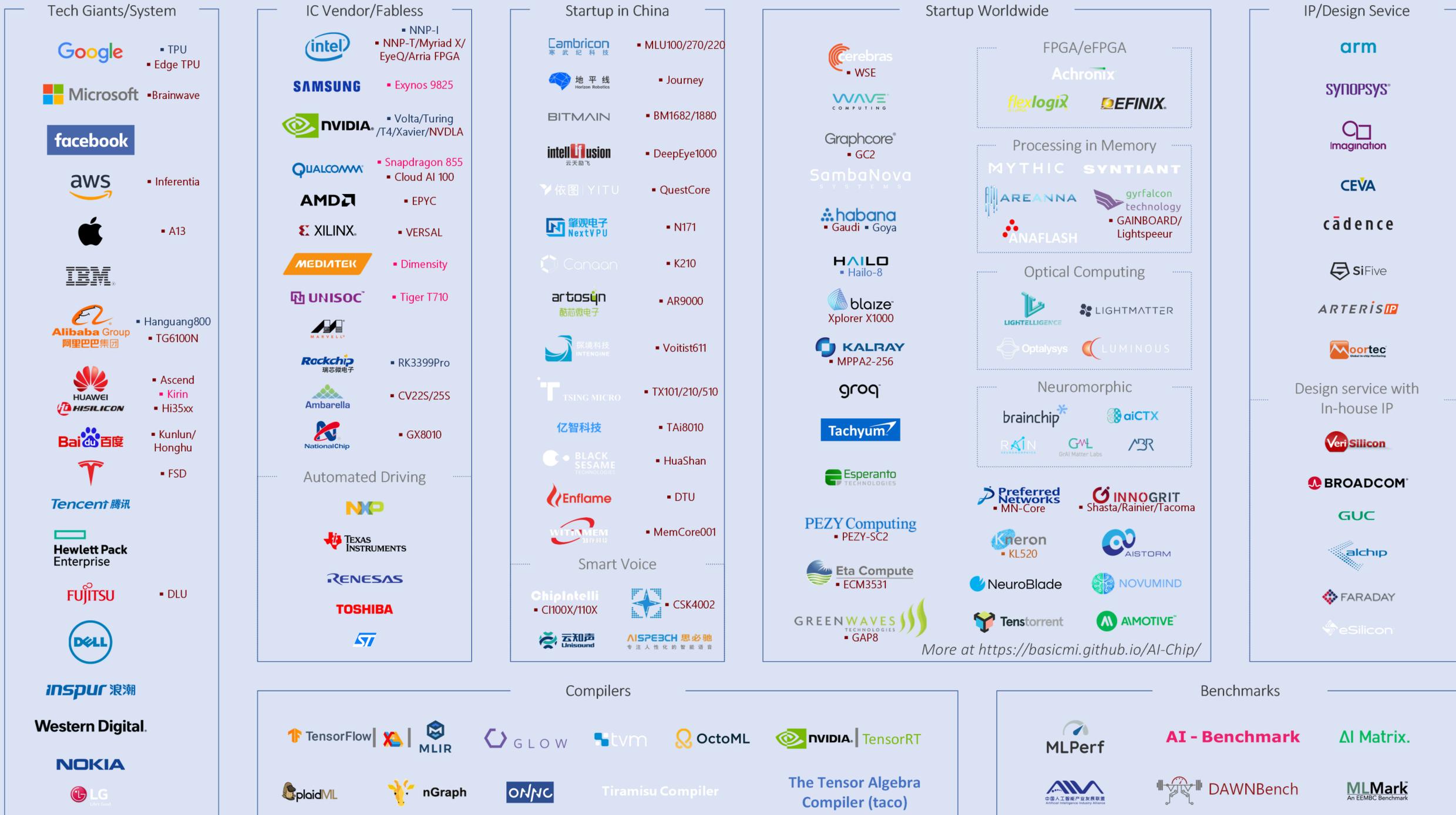
# A lot of industry activity

## AI Chip Landscape

V0.7 Dec., 2019

S.T.

- MLPerf results available
- AI-Benchmark results available



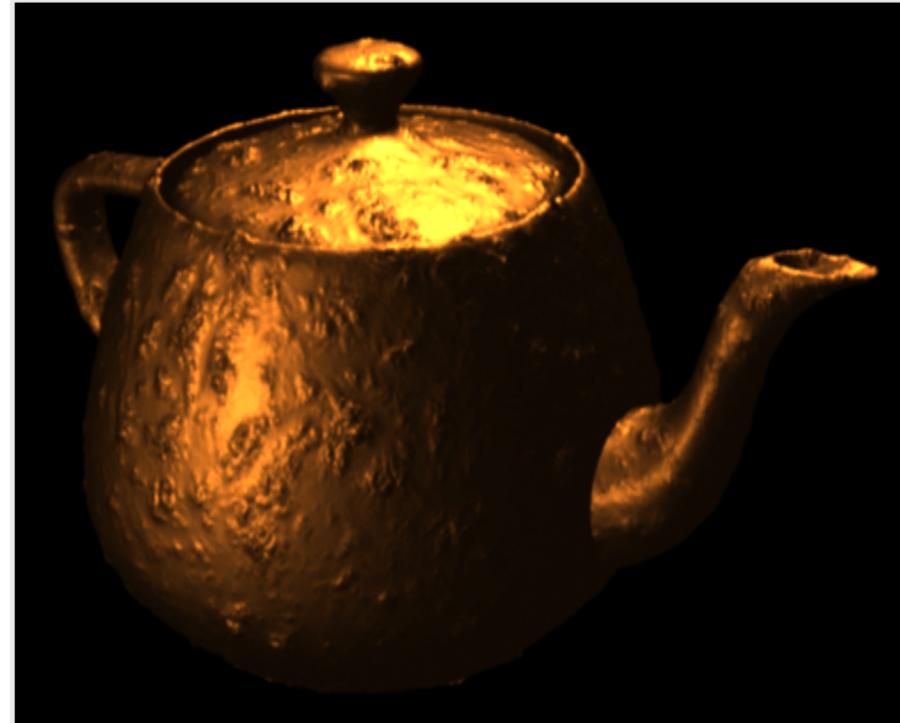
All information contained within this infographic is gathered from the internet and periodically updated, no guarantee is given that the information provided is correct, complete, and up-to-date.

# The Road to Point Reyes Lucasfilm 1984

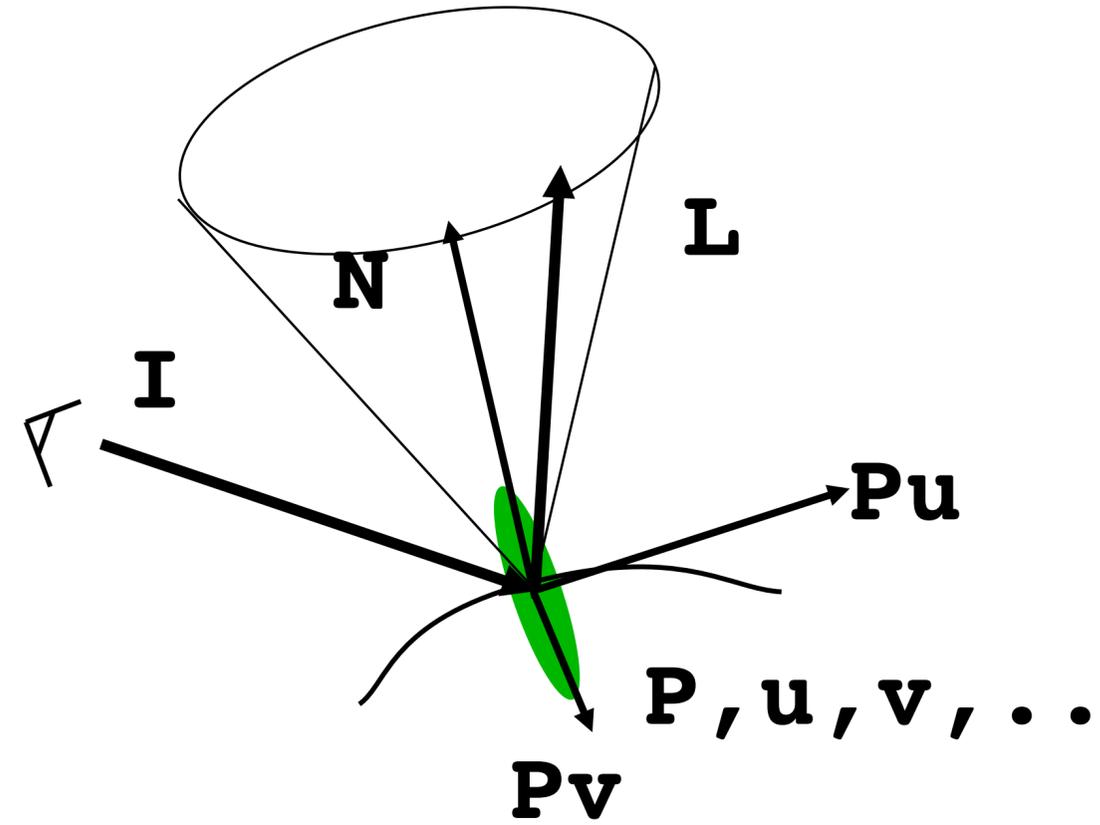
**R.E.Y.E.S = Renders Everything You Ever Saw**



```
surface corrode(float Ks=0.4, Ka=0.1, rough=0.25) {  
    float i, freq=1, turb=0;  
    // compute fractal texture  
    for( i=0; i<6; i++ ) {  
        turb+=1/freq*noise(freq*P);  
        freq*=2;  
    }  
    // perturb surface  
    P -= turb * normalize(N);  
    N = faceforward(normalize(calculatenormal(P)));  
    // compute reflection and final color  
    Ci = Cs*(Ka*ambient()+Ks*specular(N,I,rough));  
}
```



**Surface Geometry**

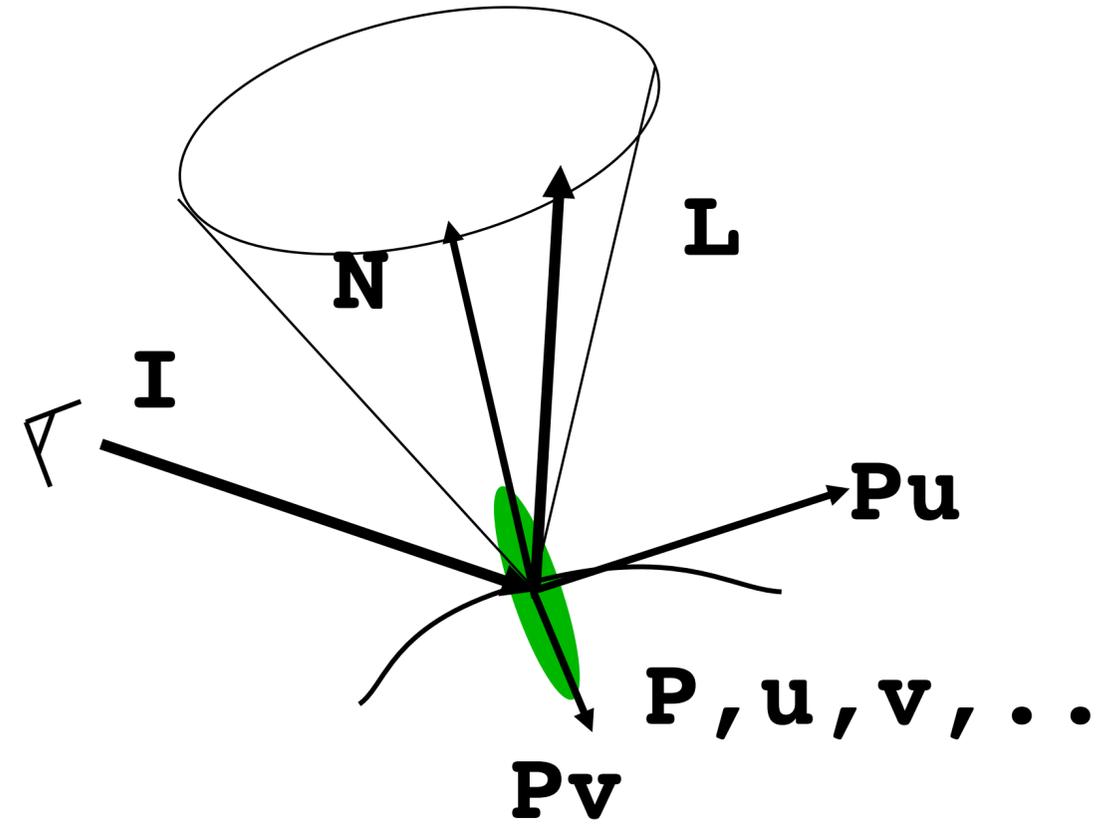


**Material**

```
Nn = normalize(N);  
illuminance( P, Nn, PI/2 ) {  
    Ln = normalize(L);  
    Ci += Cs * Cl * Ln.Nn;  
}
```

**Surface Geometry**

**Light**



illuminate(  $P, N, \text{beamangle}$  )

$$Cl = (\text{intensity} * \text{lightcolor}) / (L.L)$$

solar(  $D, 0$  )

$$Cl = \text{intensity} * \text{lightcolor};$$

# Little Languages (DSLs)

Jon Bentley, CACM 29(8), 1986

Defining “little” is harder; it might imply that the first-time user can use this system in an hour or master the language in a day, or perhaps the first implementation took just a few days. In any case, a little language is specialized to a particular problem domain and does not include many features found in conventional languages.

# UNIX "DSLs"

bash, csh - shell programming

awk - processing strings

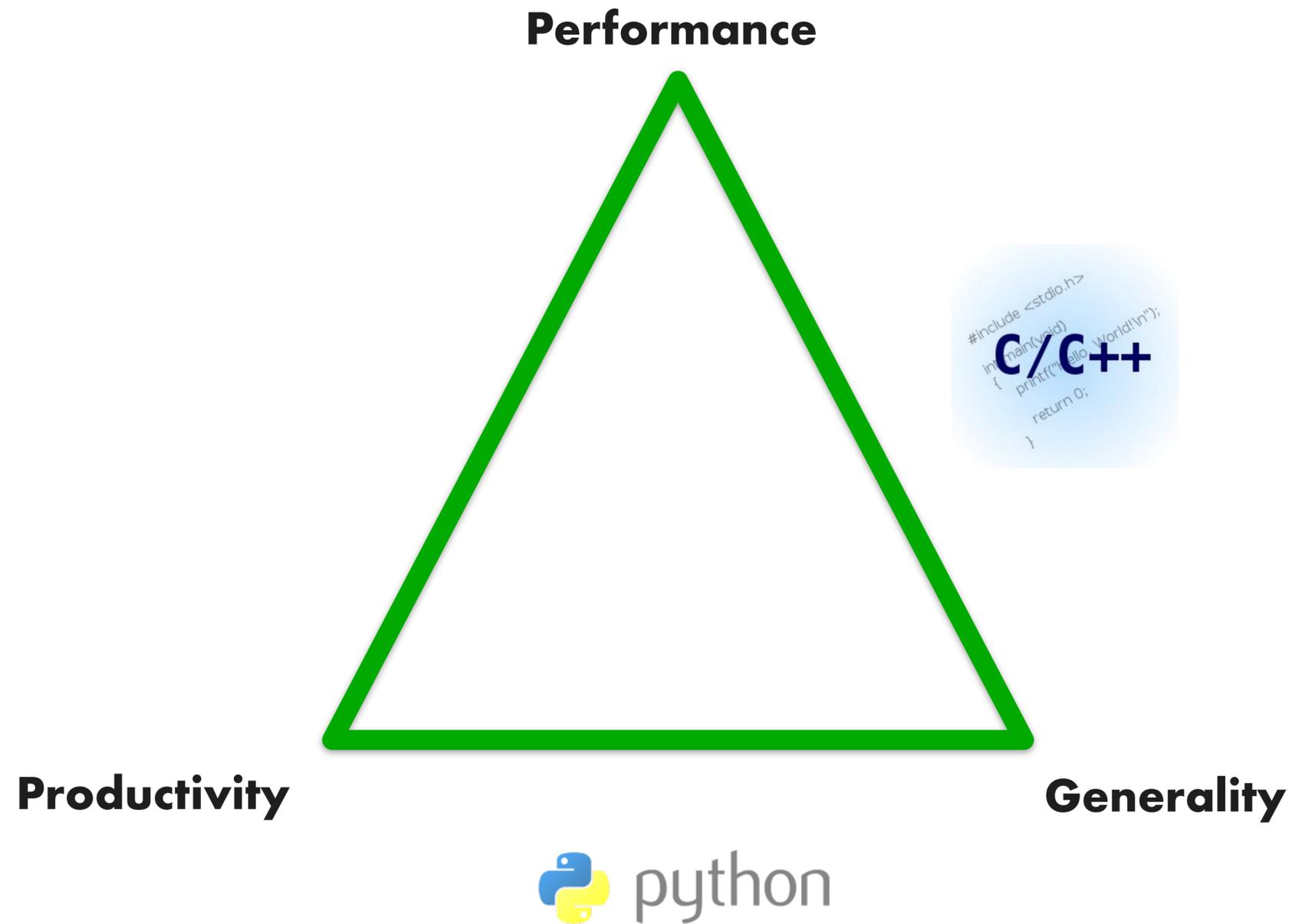
sed - regular expressions

troff, pic, tbl, eqn, ...

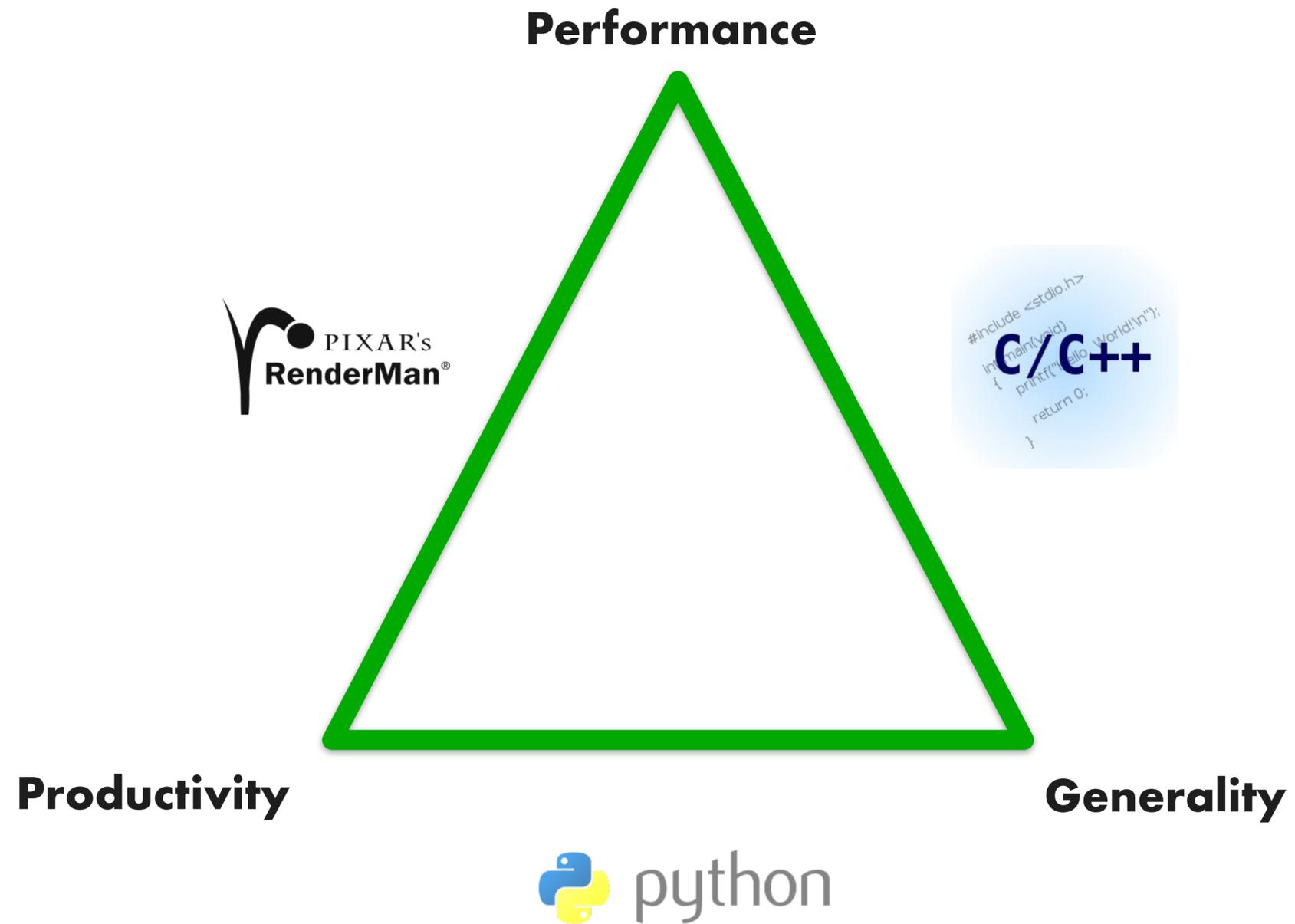
printf formatting

...

# Programming Languages



# Domain-Specific Languages



# Graphics Libraries

```
glPerspective(45.0);  
for( ... ) {  
    glTranslate(1.0,2.0,3.0);  
    glBegin(GL_TRIANGLES);  
        glVertex(...);  
        glVertex(...);  
        ...  
    glEnd();  
}  
glSwapBuffers();
```

# OpenGL “Grammar”

<Scene> = <BeginFrame> <Camera> <World>  
<EndFrame>

<Camera> = glMatrixMode(GL\_PROJECTION)  
<View>

<View> = glPerspective | glOrtho

<World> = <Objects>\*

<Object> = <Transforms>\* <Geometry>

<Transforms> = glTranslatef | glRotatef | ...

<Geometry> = glBegin <Vertices> glEnd

<Vertices> = [ glColor ] [ glNormal ] glVertex

# Advantages

## Productivity

- Graphics library is easy to use

## Portability

- Runs on wide range of GPUs

# Advantages

Productivity

Portability

Performance

- Vertices/Fragments are independent
- Rasterization can be done in hardware
- Textures are read-only; texture filtering hw
- Specialized scheduler for pipeline
- ...
- Allows for super-optimized implementations

# Advantages

Productivity

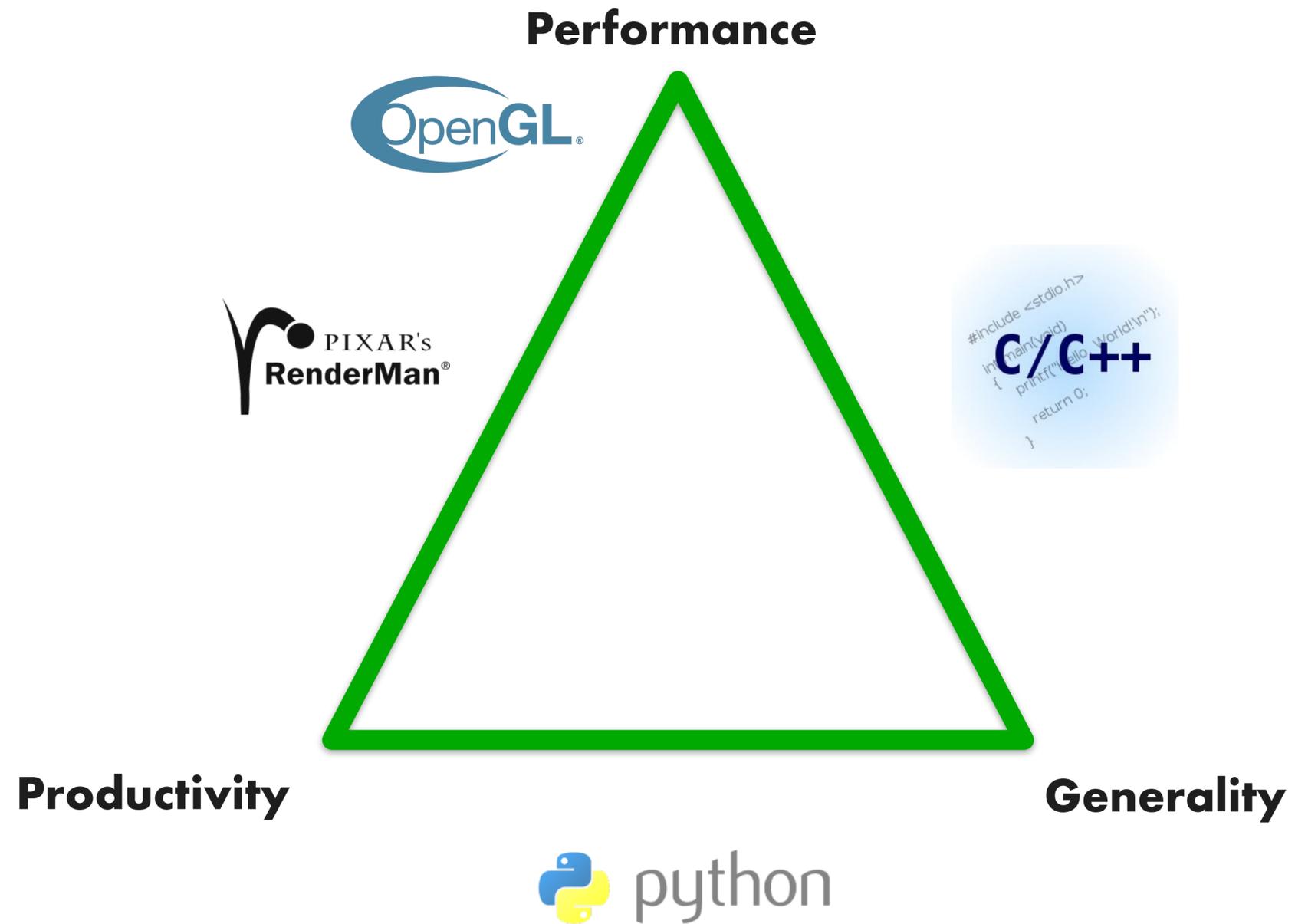
Portability

Performance

Encourage innovation

- Allows vendors to radically optimize hardware architecture to achieve efficiency
- Allows vendors to introduce new low-level programming models and abstractions

# Domain-Specific Languages



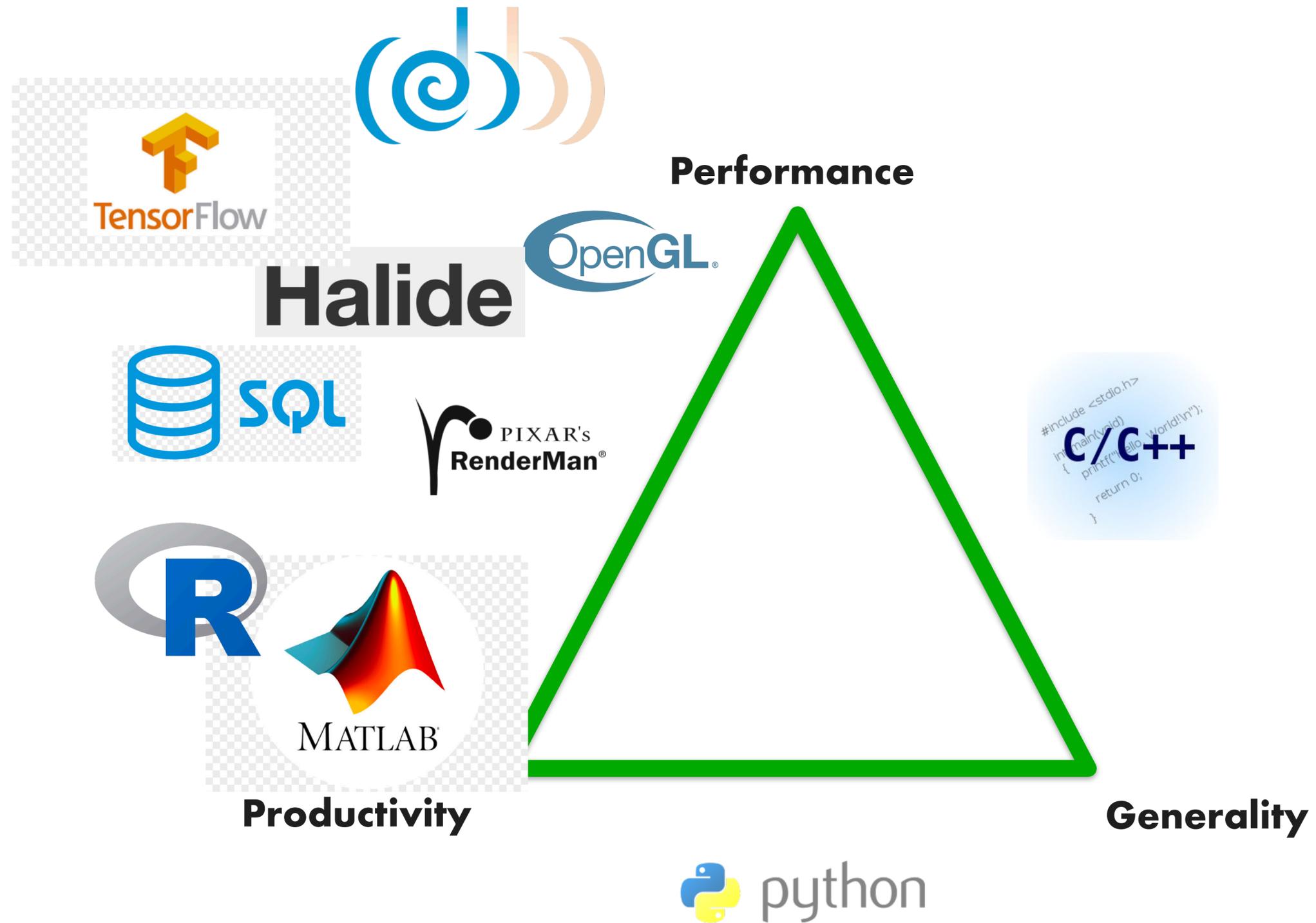
# Definition: Domain-Specific

Definition: A language or library that exploits domain knowledge for productivity and performance

Widely used in many application areas

- matlab / R
- SQL / map-reduce / Microsoft's LINQ
- TensorFlow, pytorch

# Domain-Specific Languages



# Why DSLs Work

---

## Advantages

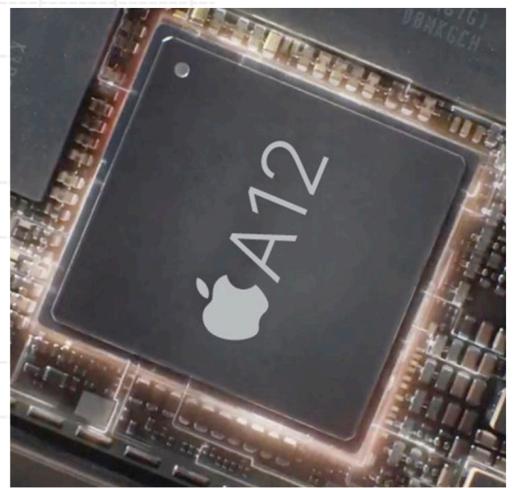
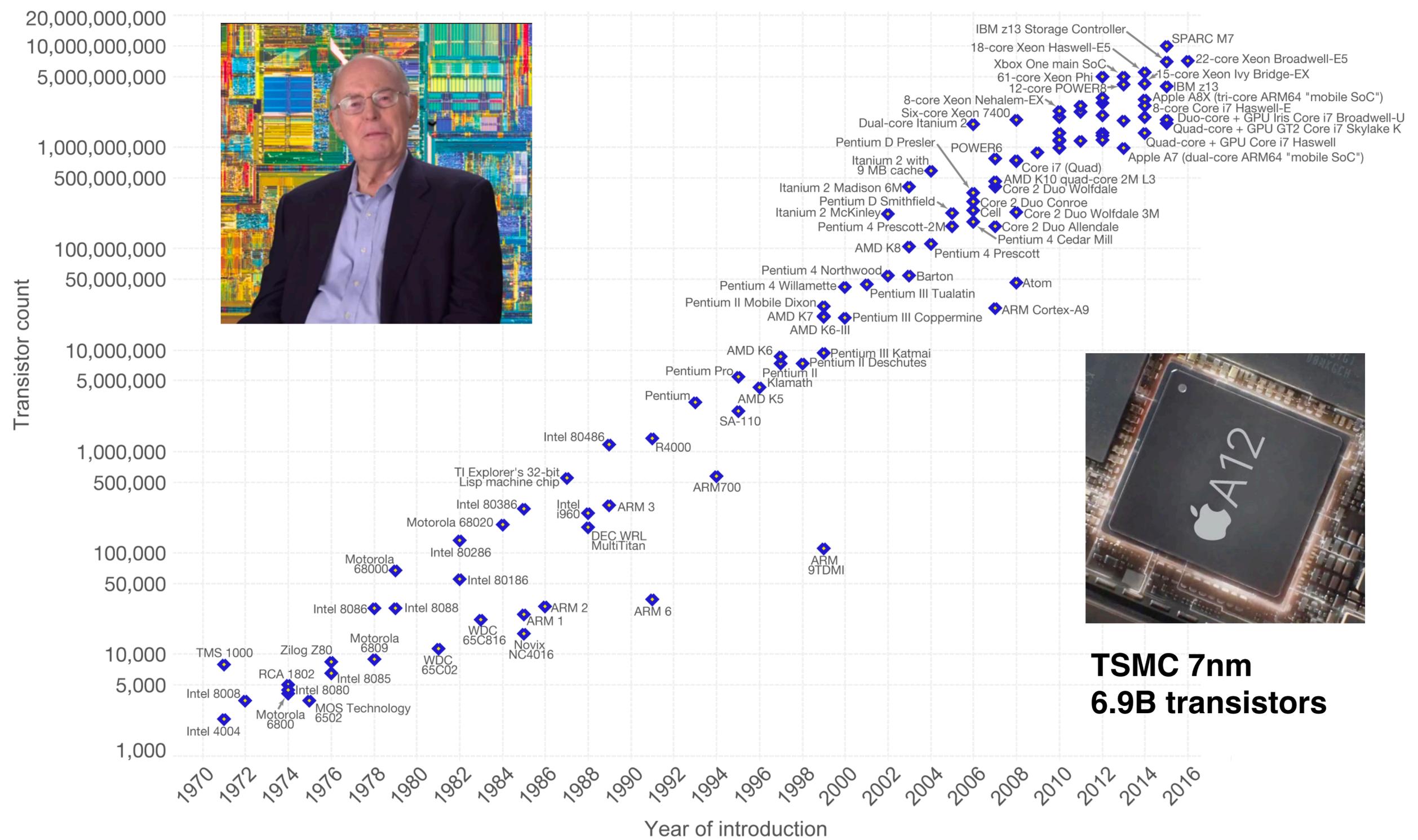
- Add the semantics of the domain
  - High-level program transformations
- Restrict programming language
  - Less-general computations
  - Guarantee static analysis
- Known parallelization strategies
  - Someone has shown how to robustly do it

=> Tractable

# Moore's Law – The number of transistors on integrated circuit chips (1971-2016)



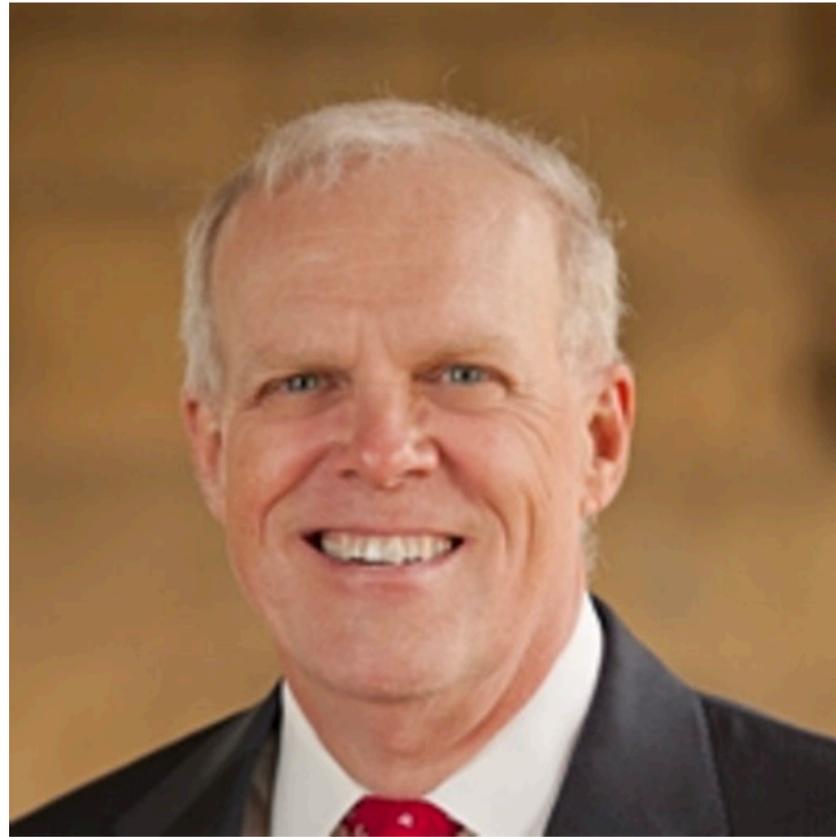
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



**TSMC 7nm**  
**6.9B transistors**

Data source: Wikipedia ([https://en.wikipedia.org/wiki/Transistor\\_count](https://en.wikipedia.org/wiki/Transistor_count))  
The data visualization is available at [OurWorldinData.org](https://www.ourworldindata.org). There you find more visualizations and research on this topic.  
Licensed under [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) by the author Max Roser.

# **A New Golden Age for Computer Architecture: Domain-Specific Hardware/Software Co-Design**



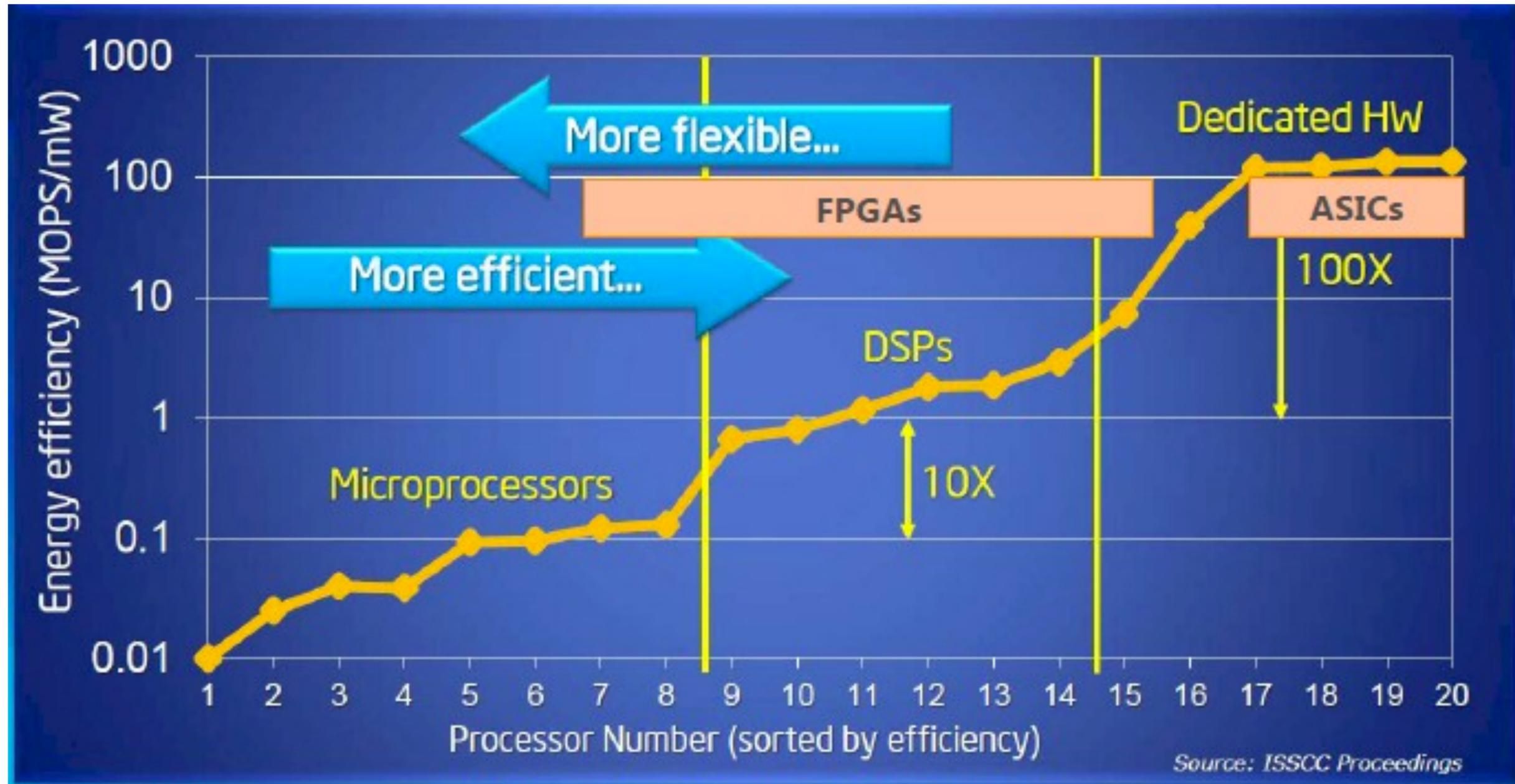
**John Hennessy**



**David Patterson**

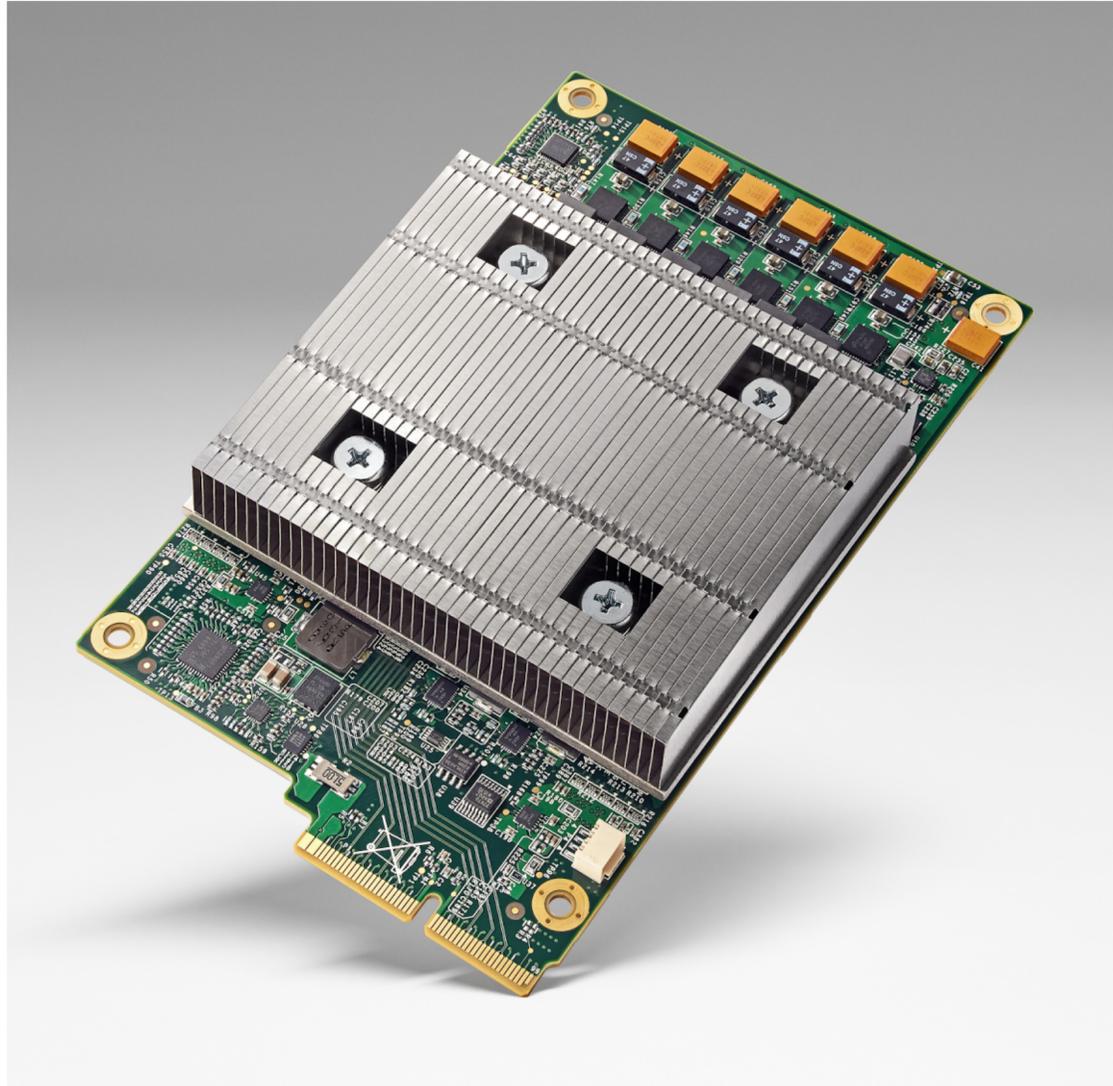
**2017 Turing Award**

# Large efficiency gains with domain-specific architectures

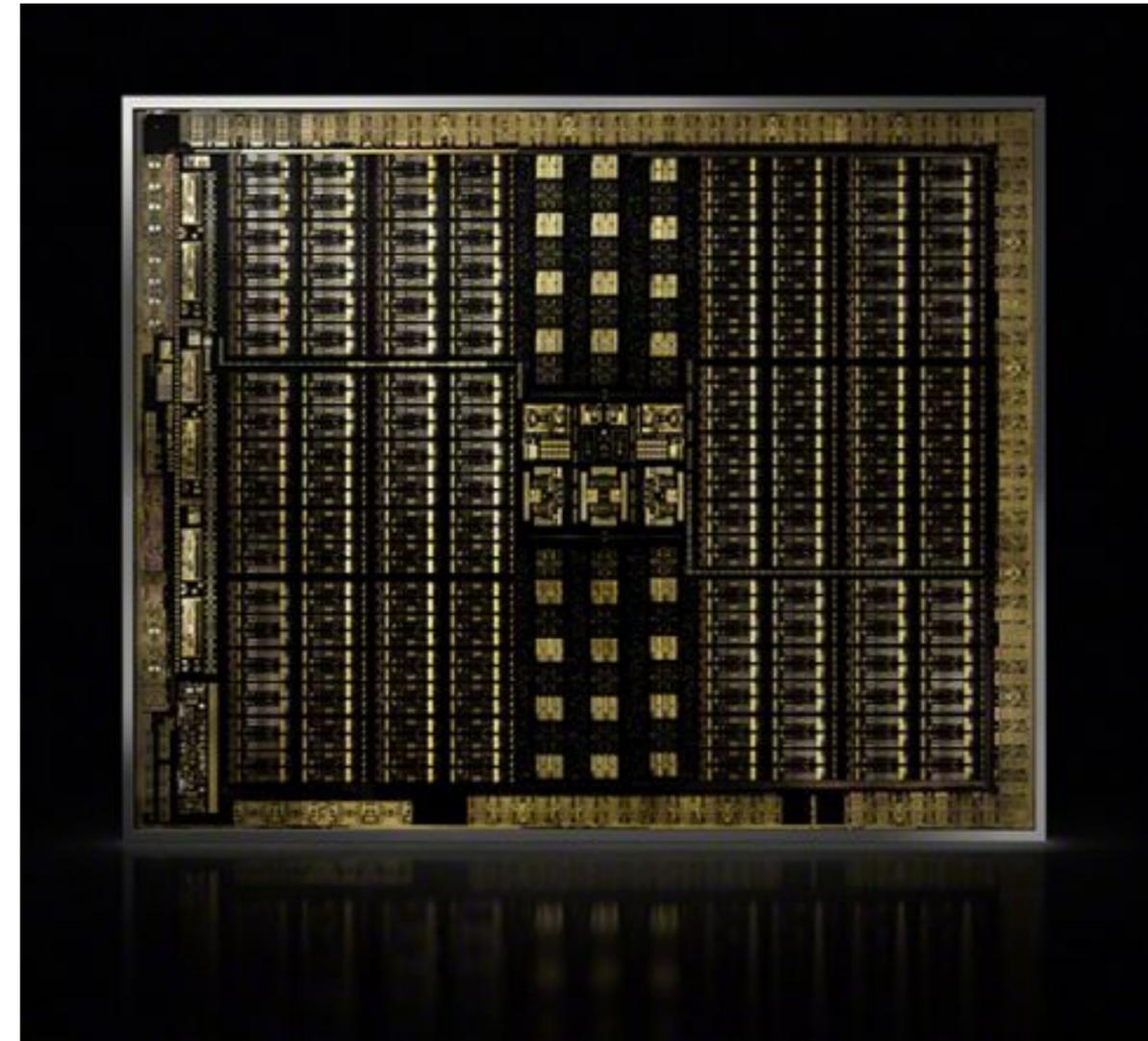


Source: Bob Broderson, Berkeley Wireless group

# Domain-Specific Architectures



**Google  
Tensor Processing Unit**



**NVIDIA  
Turing Architecture**

# Collection-Oriented Languages

## Lists

Lisp M58



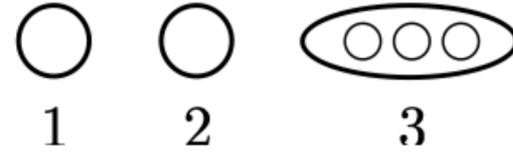
## Sets

SETL S70



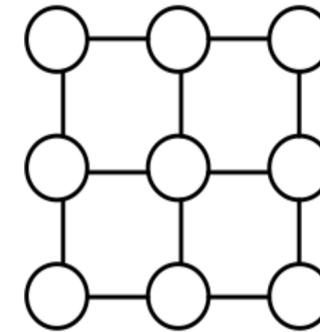
## Nested Sequences

NESL B94



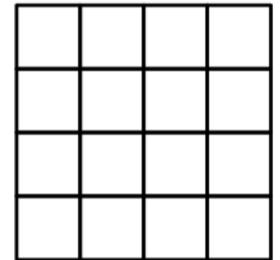
## Grids

Sejits S09, Halide



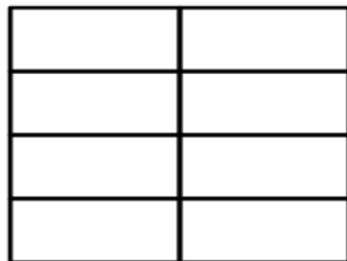
## Arrays

APL I62



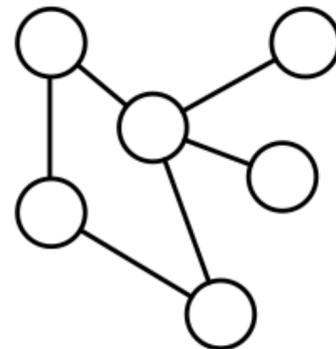
## Relations

Relational Algebra C70,



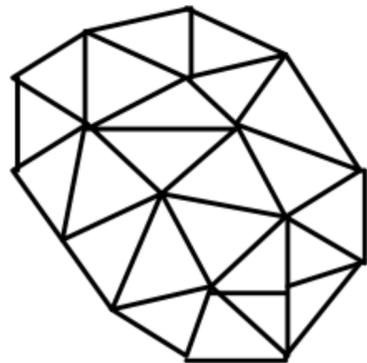
## Graphs

GraphLab L10



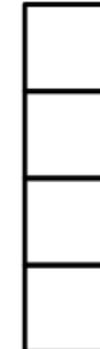
## Meshes

Liszt D11



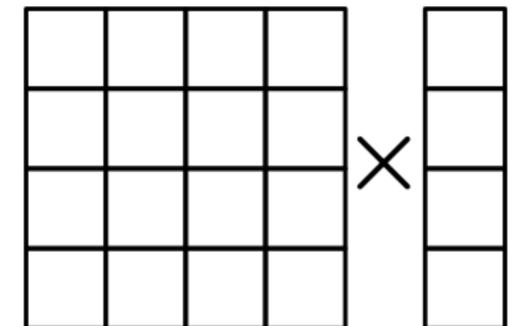
## Vectors

Vector Model B90



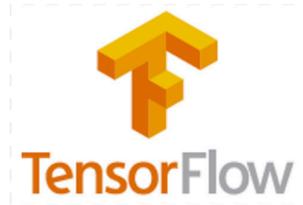
## Matrices and Tensors

Matlab M79, taco K17



A collection-oriented programming model provides collective operations on some collection/abstract data structure

# Modern Domain-Specific Languages/Compilers



**Halide**



# Lecture Overview

