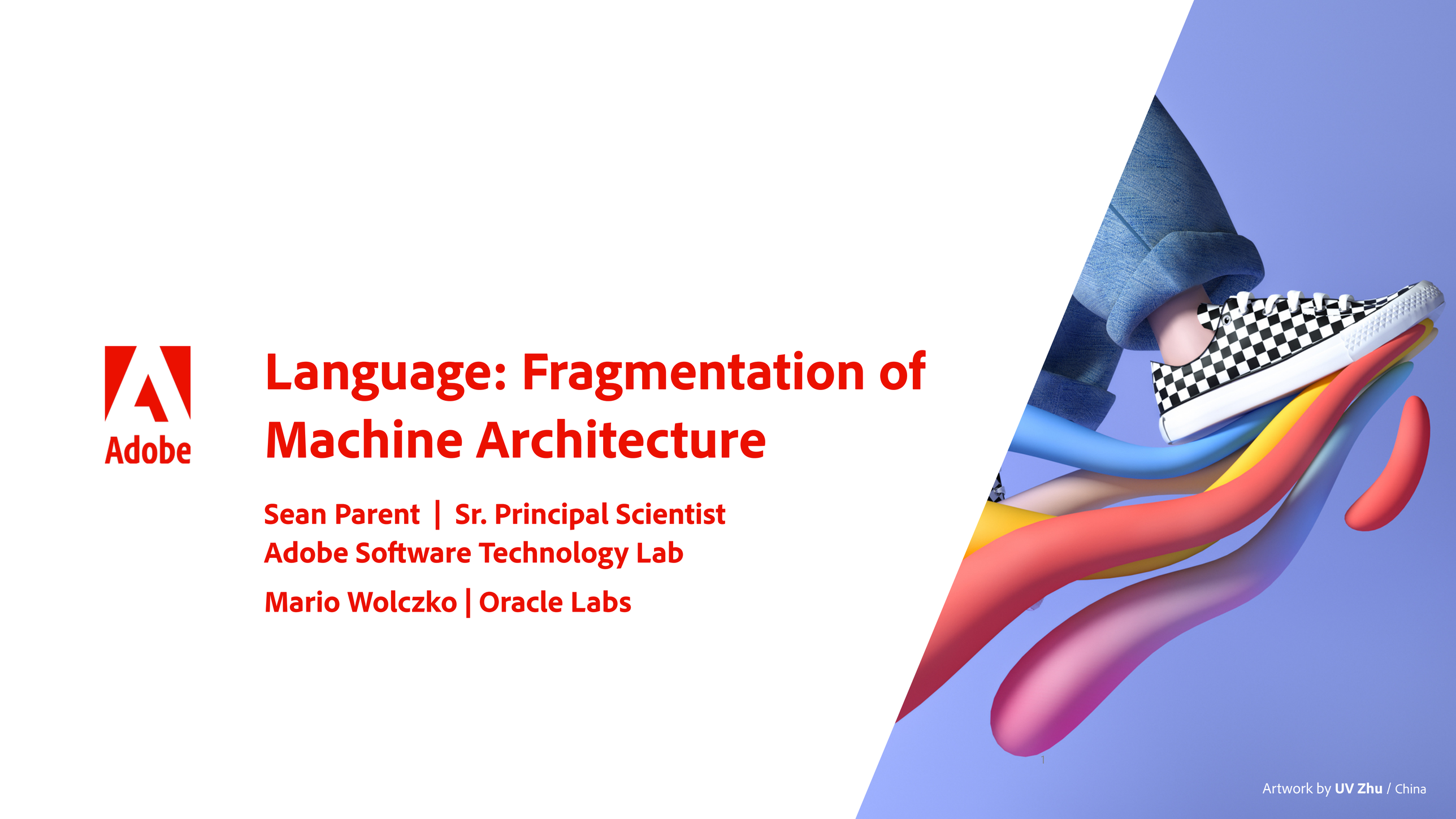


Language: Fragmentation of Machine Architecture

Sean Parent | Sr. Principal Scientist
Adobe Software Technology Lab

Mario Wolczko | Oracle Labs

Artwork by **UV Zhu** / China

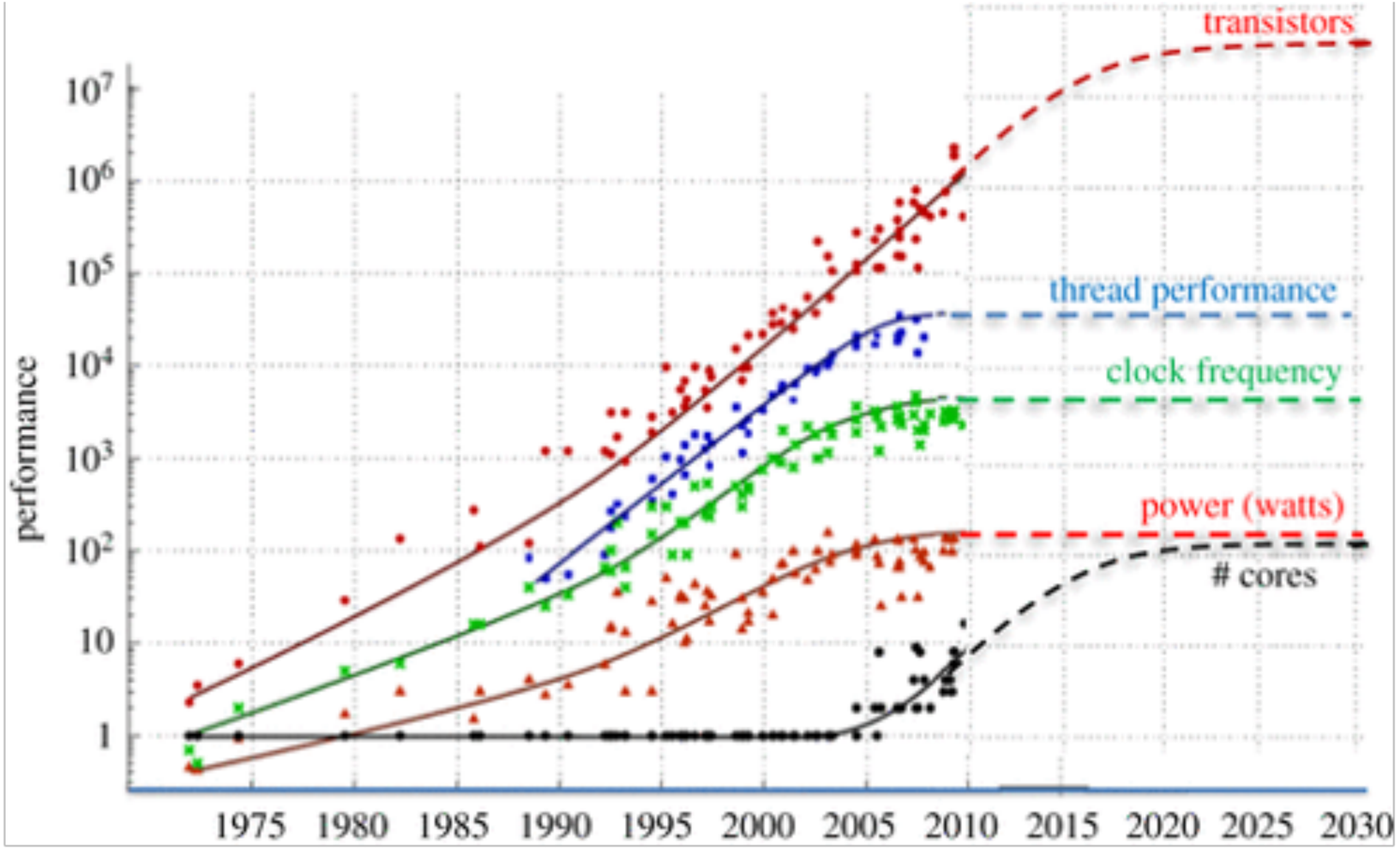
Desktop Applications – Recent History

- Macintosh
  - 68K, single-core
  - PPC, single-core
  - Intel, multi-core, SIMD, OpenGL/CL
- Windows
  - Intel, single-core
  - Intel, multi-core, SIMD, OpenGL/CL

Two Key Events

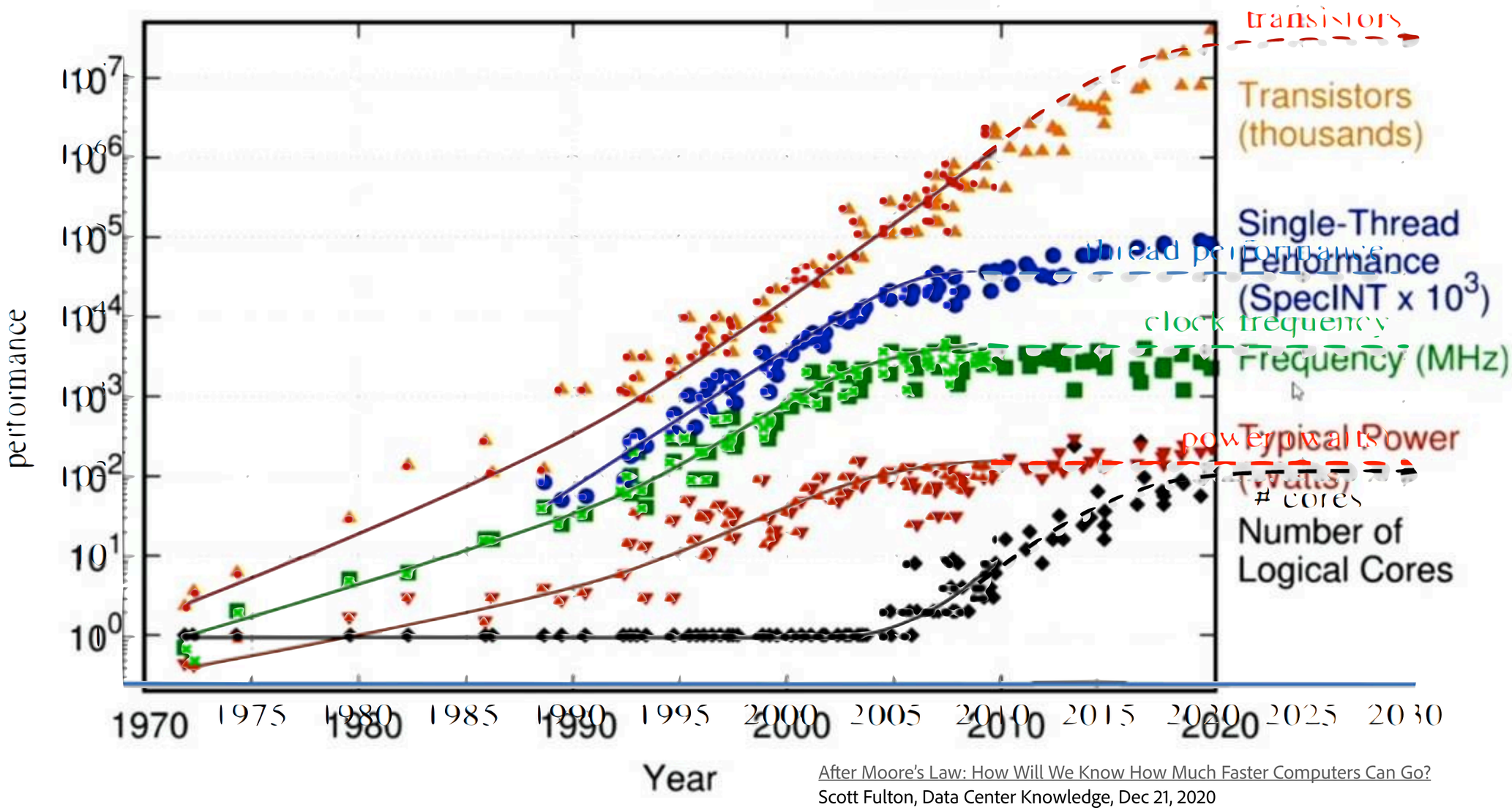
- 2005 we hit the physical limits of Moore's Law under current technology
- 2007 the iPhone is introduced

2009 Projected Processor Characteristics



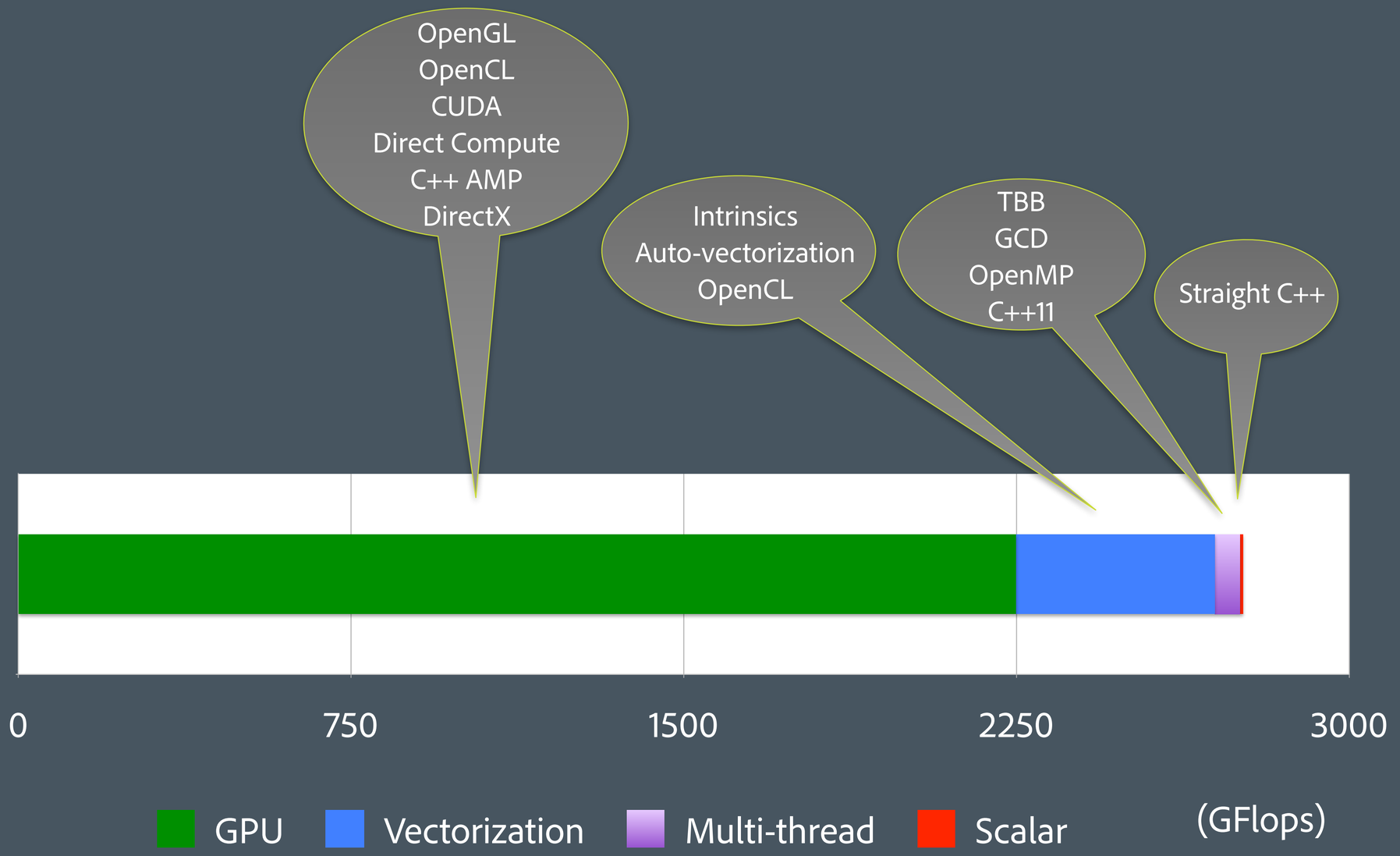
The future of computing beyond Moore's Law, Volume: 378, Issue: 2166, DOI: (10.1098/rsta.2019.0061)

How are we doing?



After Moore's Law: How Will We Know How Much Faster Computers Can Go?
Scott Fulton, Data Center Knowledge, Dec 21, 2020

Desktop Compute Power (8-core 3.5GHz Sandy Bridge + AMD Radeon 6950)



Adobe

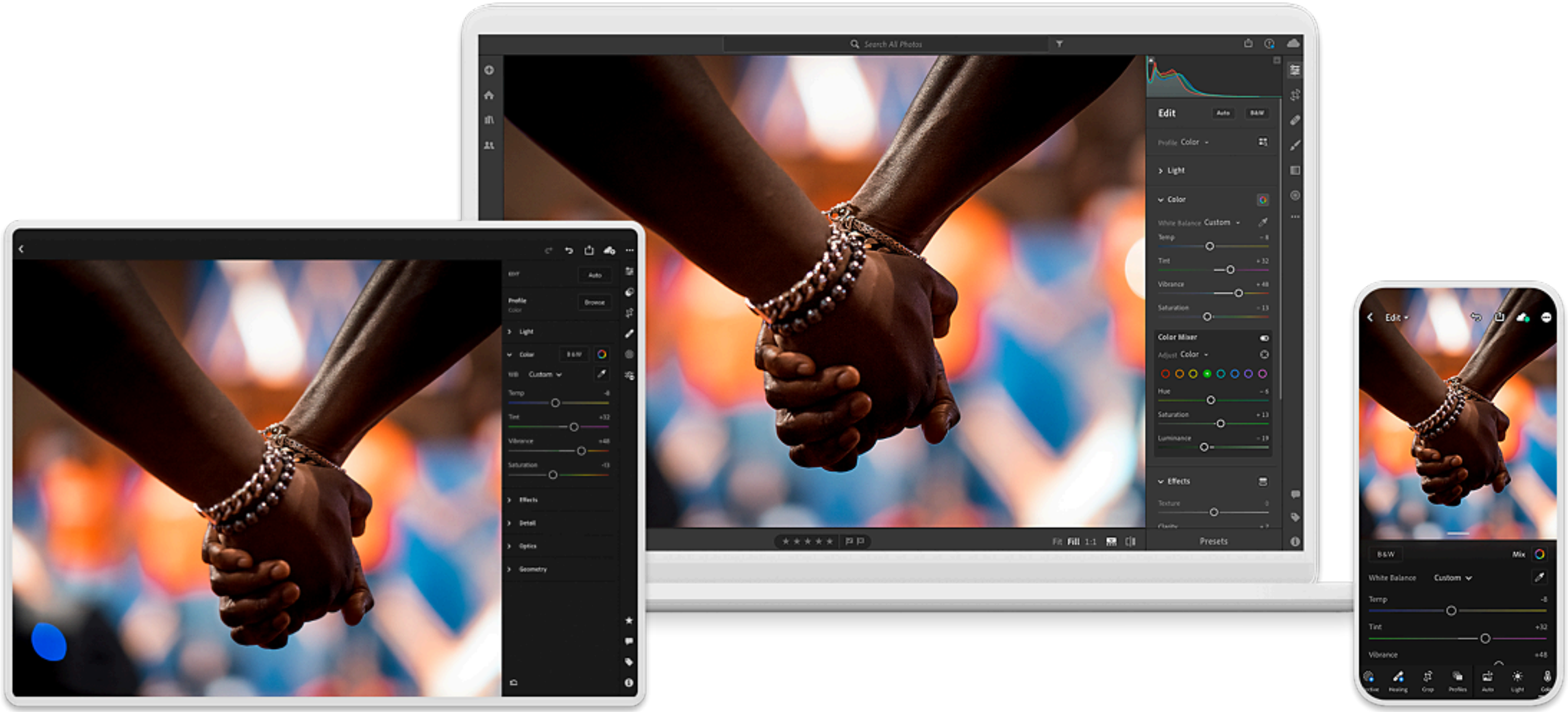
Platform Expansion

- Mobile
  - iPhone fundamentally changed mobile devices
- Web
  - *Content Ubiquity* is expected
- Tablets
  - Larger “phones” succeeded where smaller desktops failed

Platform Expansion

- In 2012 I gave an internal presentation at Adobe on content ubiquity
  - Broadband was available to th majority of the popultion in the developed countries
    - Soon will be true worldwide
  - Noted capabilities of mobile devices
    - Increased by > 8x
  - Because content ubiquity is becoming a base expectation, not providing it will kill a product

Platform Expansion



Platform Fragmentation

- macOS
- Win32 & UXP
- iOS and iPadOS
- Android
- Linux (server)
- W3C

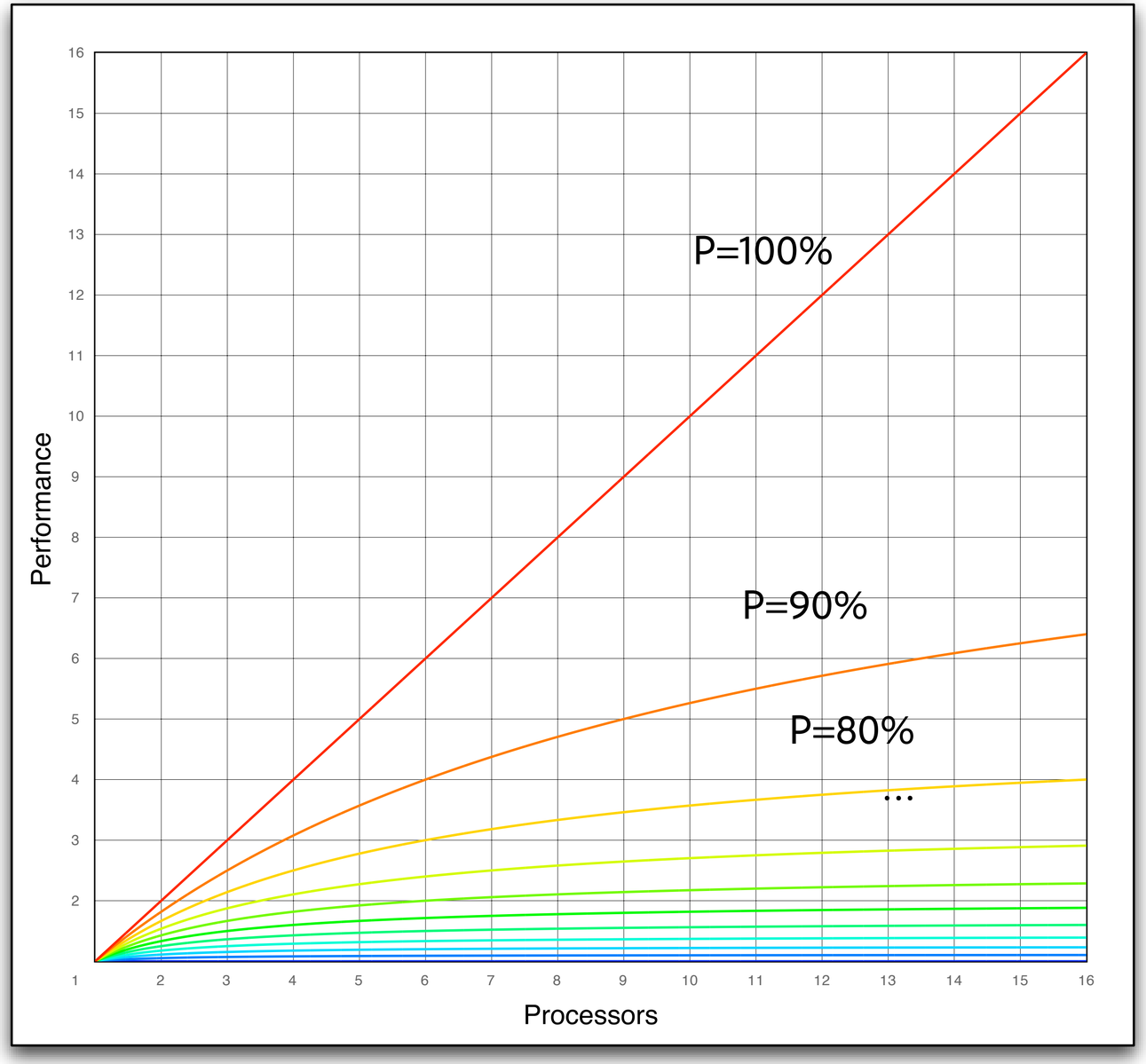
Instruction Set Fragmentation

- Intel (AVX SIMD)
- ARM (Neon SIMD)
- WASM (WASM SIMD)
  - Currently 32 bit address space

GPU Platform Fragmentation

- Metal (Apple)
- DX12 (Microsoft)
- Vulcan (Open Standard, Android, Linux)
- CUDA (NVIDIA)
- WebGPU (Browsers)

Amdahl's Law



Hardware to Fight Amdahl's Law

- NUMA
- DMA to discrete GPU
- Unified Memory (Apple's M1 chips)
  - "The unified memory requires a very different approach to that on Windows with discrete GPUs."
- Optane?

Hardware to Manage Power

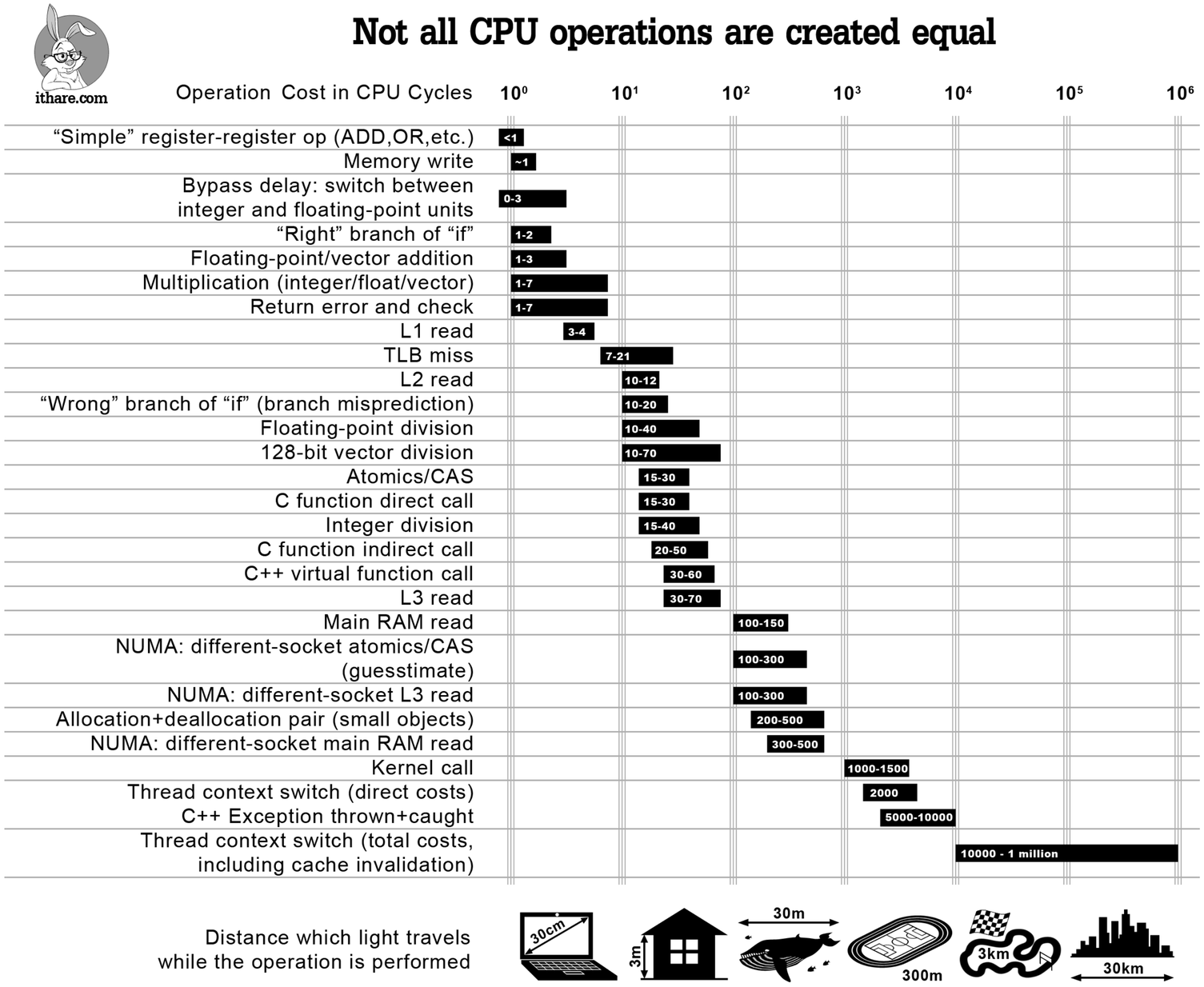
- Thermal Throttling
- Heterogeneous Cores
- Discrete / Integrated GPU Switching

Languages Are Not Keeping Up

- We are struggling to find models to reason about concurrent systems
  - CSP, Actors, Functional, …
- Safer languages have higher overhead
  - But unsafe languages are harder to get correct

- My estimate is we are leaving 2^3 to 2^5 times performance on the table

Operation Costs are Not Reflected In Code



What is wrong with C++

- C++ allow us to control memory layout and sharing
- Compiler is blind to sharing - *aliasing* + *mutation* kills optimization
  - Developer is also blind to sharing making code difficult to reason about
- Lack of safety makes it very difficult especially in the presence of concurrency for new developers
- Basic library primitives for concurrency (threads) are very expensive
- Performance penalty (Stepanov Abstraction Penalty) to wrap basic arithmetic types
  - i.e. treat a uint8_t as a value from 0.0 - 1.0
  - Code the same algorithm with different function names or pay the tax

C++

- Despite the limitations and drawbacks, C++ is still performance king*
- Adobe has a massive investment in C++ code bases
- Can C++ be improved *enough?*
  - The pace of C++ advancement is still rapid
- If another language proved to be better, what does the migration look like?

Possible Future?

- In 2007 I gave a Google TechTalk, A Possible Future of Software Development
  - Observation - Most developers cannot write a correct binary search (still true)
    - An argument for developing generic libraries and concepts
  - Conjecture - All problems of scale become a network problem
  - An argument for developing declarative systems
    - BNF, SQL, HTML, Spreadsheets
  - In imperative languages a single relationship becomes multiple functions

Imperative Solution to Mini-Image Size

```
#import "ImageSizeController.h"
#import <Foundation/NSObject.h>
#import <AppKit/NSNibDeclarations.h>
#import <AppKit/NSControl.h>
#import <AppKit/NSCell.h>
#import <Foundation/NSNumberFormatter.h>
#import <Foundation/NSNotification.h>
#import <AppKit/NSTextField.h>
#import <math.h>
#import <stddef.h>


/* Reading a text field with a formatter attached forces
the text through the formatter. */

static double TextField_unformattedDoubleValue(
                  id textField ) {
    id formatter = [ textField formatter ];
    [ textField setFormatter: nil ];
    double result = [ textField doubleValue ];
    [ textField setFormatter: formatter ];
    return result;
    }


/* Same logic but for integers. */

static double TextField_unformattedIntValue(
                  id textField ) {
    id formatter = [ textField formatter ];
    [ textField setFormatter: nil ];
    int result = [ textField intValue ];
    [ textField setFormatter: formatter ];
    return result;
    }


/* Setting a text field while it is editing doesn't manage
to set the text. So, we have to stop editing and the
restart. */

static void TextField_setDoubleValueAndFormatter(
                  id textField,
                  double value,
                  NSFormatter *formatter ) {
    BOOL wasEditing = [ textField abortEditing ];

    /* If we're changing the formatter, then we want to
    make sure that the display updates including the edit
    field. */

    if( [ textField formatter ] != formatter ) {
        [ textField setFormatter: formatter ];
        [ textField setDoubleValue: value - 1.0 ];
    }

    [ textField setDoubleValue: value ];

    if( wasEditing ) {
        [ textField selectText: nil ];
    }
}


/* Same logic but with integer values. */

static void TextField_setIntValueAndFormatter(
                  id textField,
                  int value,
                  NSFormatter *formatter ) {
    BOOL wasEditing = [ textField abortEditing ];

    /* If we're changing the formatter, then we want to
    make sure that the display updates including the edit
    field. */

    if( [ textField formatter ] != formatter ) {
        [ textField setFormatter: formatter ];
        [ textField setIntValue: value - 1 ];
    }

    [ textField setIntValue: value ];

    if( wasEditing ) {
        [ textField selectText: nil ];
    }
}
```

```
/* Here is the class declaration for the controller. */

@interface ImageSizeController : NSObject {
    IBOutlet id heightField_;
    IBOutlet id widthField_;
    IBOutlet id constrainProportionsBox_;
    IBOutlet id usePercentagesBox_;
    IBOutlet NSNumberFormatter *pixelFormatter_;
    IBOutlet NSNumberFormatter *percentFormatter_;
@private
    int initialWidthPixels_;
    int initialHeightPixels_;
    int widthPixels_;
    int heightPixels_;
    double widthPercentage_;
    double heightPercentage_;
    BOOL constrainProportions_;
    BOOL usePercentages_;
}
- (void) showWidth;
- (void) showHeight;
- (void) showAll;
- (IBAction) heightAction: (id)sender;
- (IBAction) widthAction: (id)sender;
- (IBAction) constrainProportionsAction: (id)sender;
- (IBAction) usePercentagesAction: (id)sender;
- (IBAction) apply: (id)sender;
- (IBAction) revert: (id)sender;
- (void) awakeFromNib;
@end


@implementation ImageSizeController

/* Update the width field. */

- (void) showWidth {
    if( usePercentages_ ) {
        TextField_setDoubleValueAndFormatter(
            widthField_, widthPercentage_,
            percentFormatter_ );
    } else {
        TextField_setIntValueAndFormatter(
            widthField_, widthPixels_, pixelFormatter_ );
    }
}


/* Update the height field. */

- (void) showHeight {
    if( usePercentages_ ) {
        TextField_setDoubleValueAndFormatter(
            heightField_, heightPercentage_,
            percentFormatter_ );
    } else {
        TextField_setIntValueAndFormatter(
            heightField_, heightPixels_, pixelFormatter_ );
    }
}


/* Update width and height fields. */

- (void) showWidthAndHeight {
    [ self showWidth ];
    [ self showHeight ];
}


/* Update all controls. */

- (void) showAll {
    [ self showWidthAndHeight ];
    [ usePercentagesBox_ setState:
          usePercentages_ ? NSOnState : NSOffState ];
    [ constrainProportionsBox_ setState:
          constrainProportions_ ? NSOnState : NSOffState ];
}


/* Revert the width and height. This works regardless of
the checkbox states. */

- (void) revertWidthAndHeight {
    widthPixels_ = initialWidthPixels_;
    widthPercentage_ = 100.0;

    heightPixels_ = initialHeightPixels_;
    heightPercentage_ = 100.0;

    [ self showWidthAndHeight ];
}


/* The revert button does its work via
revertWidthAndHeight. */

- (IBAction) revert: (id) sender {
    [ self revertWidthAndHeight ];
}


/* Handle the apply button by copying over the width and
height. This also sets the percentage values. If we are
displaying percentages, then we need to update. We update
for pixels as well in case this forced any rounding. */

- (IBAction) apply: (id) sender {
    initialWidthPixels_ = widthPixels_;
    widthPercentage_ = 100.0;

    initialHeightPixels_ = heightPixels_;
    heightPercentage_ = 100.0;

    [ self showWidthAndHeight ];
}


/* Handle an event from the use percentages checkbox. */

- (IBAction) usePercentagesAction: (id) sender {
    BOOL newUsePercentages = [ sender state ] == NSOnState;
    if( newUsePercentages != usePercentages_ ) {
        usePercentages_ = newUsePercentages;
        [ self showWidthAndHeight ];
    }
}


/* Handle an event from the constrain proportions checkbox.
*/

- (IBAction) constrainProportionsAction: (id) sender {
    BOOL newConstrainProportions =
            [ sender state ] == NSOnState;
    if( newConstrainProportions != constrainProportions_ ){
        constrainProportions_ = newConstrainProportions;
        if( newConstrainProportions ) {
            [ self revertWidthAndHeight ];
        }
    }
}


/* The following routines handle conversion between pixels
and percentages for width and height. */

- (void) widthPixelsFromPercentage {
    widthPixels_ = (int)
        floor( initialWidthPixels_ * widthPercentage_
                    / 100.0 + 0.5 );
}


- (void) widthPercentageFromPixels {
    widthPercentage_ =
        widthPixels_ * 100.0 / initialWidthPixels_;
}


- (void) heightPixelsFromPercentage {
    heightPixels_ = (int)
        floor( initialHeightPixels_ * heightPercentage_
                    / 100.0 + 0.5 );
}


- (void) heightPercentageFromPixels {
    heightPercentage_ =
        heightPixels_ * 100.0 / initialHeightPixels_;
}


/* Process a change to the width field. */

- (IBAction) widthAction: (id) sender {
    if( usePercentages_ ) {
        widthPercentage_ =
            TextField_unformattedDoubleValue( sender );
        [ self widthPixelsFromPercentage ];
    } else {
        widthPixels_ =
            TextField_unformattedIntValue( sender );
        [ self widthPercentageFromPixels ];
    }

    if( constrainProportions_ ) {
        heightPercentage_ = widthPercentage_;
        [ self heightPixelsFromPercentage ];
        [ self showHeight ];
    }
}


/* Process a change to the height field. */

- (IBAction) heightAction: (id) sender {
    if( usePercentages_ ) {
        heightPercentage_ =
            TextField_unformattedDoubleValue( sender );
        [ self heightPixelsFromPercentage ];
    } else {
        heightPixels_ =
            TextField_unformattedIntValue( sender );
        [ self heightPercentageFromPixels ];
    }

    if( constrainProportions_ ) {
        widthPercentage_ = heightPercentage_;
        [ self widthPixelsFromPercentage ];
        [ self showWidth ];
    }
}


/* Trigger the text field actions in response to actual
changes. */

- (void) controlTextDidChange:
            (NSNotification *) notification {
    id sender = [ notification object ];
    SEL action = [ sender action ];
    if( action ) {
        [ [ sender target ]
                performSelector: action
                withObject: sender ];
    }
}


/* When we start up, we want to set initial values. This
would ordinarily be
done by code that was creating the controller and then
running it with the dialog
NIB, but we aren't worrying about that here. */

- (void) awakeFromNib {
    initialWidthPixels_ = widthPixels_ = 400;
    initialHeightPixels_ = heightPixels_ = 300;
    widthPercentage_ = 100.0;
    heightPercentage_ = 100.0;
    constrainProportions_ = YES;
    usePercentages_ = NO;
    [ self showAll ];
}

@end
```

Declarative Solution using the Property Model Library

```
sheet mini_image_size
{
 input:
        original_width   : 5 * 300;
        original_height  : 7 * 300;
 interface:
        constrain        : true;
        width_pixels     : original_width      <== round(width_pixels);
        height_pixels    : original_height     <== round(height_pixels);
        width_percent;
        height_percent;
 logic:
        relate {
            width_pixels      <== round(width_percent * original_width / 100);
            width_percent     <== width_pixels * 100 / original_width;
        }
        relate {
            height_pixels     <== round(height_percent * original_height / 100);
            height_percent    <== height_pixels * 100 / original_height;
        }
        when (constrain) relate {
            width_percent     <== height_percent;
            height_percent    <== width_percent;
        }
 output:
        result <== { height: height_pixels, width: width_pixels };
}
```

Where do programming languages need to go

- Major shift from developer productivity to code efficiency
- Locality, locality, locality
  - Data oriented, array based
- Value semantics with safe mutability
  - Reference semantics and garbage collectors are problematic
- Computation kernels
  - Supporting SIMD and GPU code generation
  - See Halide language

Example of Halide

```
Func blur_3x3(Func input) {
  Func blur_x, blur_y;
  Var x, y, xi, yi;

  // The algorithm - no storage or order
  blur_x(x, y) = (input(x-1, y) + input(x, y) + input(x+1, y))/3;
  blur_y(x, y) = (blur_x(x, y-1) + blur_x(x, y) + blur_x(x, y+1))/3;

  // The schedule - defines order, locality; implies storage
  blur_y.tile(x, y, xi, yi, 256, 32)
        .vectorize(xi, 8).parallel(y);
  blur_x.compute_at(blur_y, x).vectorize(x, 8);

  return blur_y;
}
```

Possible?

```
blur_3x3 = [2] => (([ -1, 0 ] + [ 0, 0 ] + [ 1, 0 ]) / 3) |
                  (([ 0, -1 ] + [ 0, 0 ] + [ 0, 1 ]) / 3);
```

Where do programming languages need to go

- Switch emphasis from safety to correctness
  - Higher level semantics allows for more optimization
- Graph based
  - Ability to control flow between software components
- Shift from functions to relationships

Machine Learning - the wild card

- CoreML (Apple)
- DirectML (Microsoft)
- Neural Engine (Apple)
- TPU (Google)

About the artist

UV Zhu

With an eye for the abstract, Chinese artist UV Zhu remixes patterns, textures, and colors to explore the future of fashion. Using Adobe Photoshop, Adobe Illustrator, and Maxon Cinema 4D, he blends surreal settings, organic shapes, and even favorite foods to challenge convention. Inspired by his travels—around the Internet and in real life—for this piece, UV fantasized about characters moving through an imaginary world, the things they might do, and what they might wear. The result is a bright, colorful expression of joy and positivity.

Made with

Adobe Photoshop Adobe Illustrator

Artwork by **UV Zhu** / China