# Microsoft Research

Each year Microsoft Research hosts hundreds of influential speakers from around the world including leading scientists, renowned experts in technology, book authors, and leading academics, and makes videos of these lectures freely available.

```
#if ABSOLUTE
  position = new;
#elif RELATIVE
  position += new;
#endif
```


# Why don't we have both?

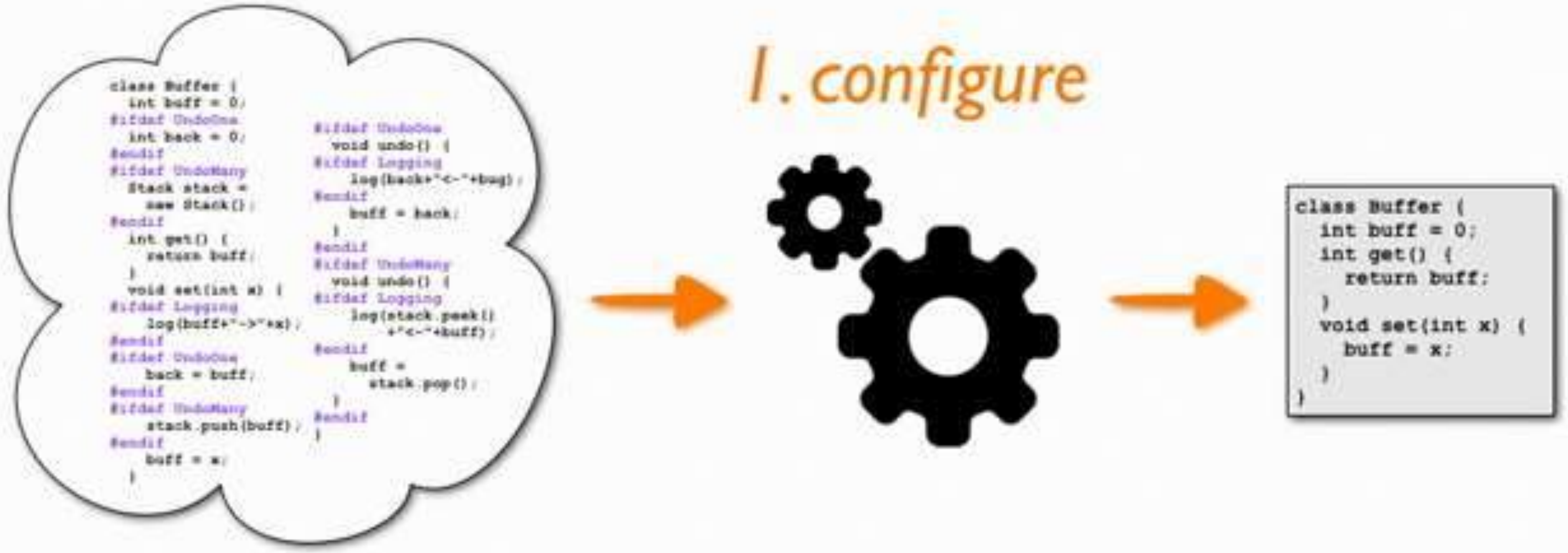
## Applications of variational programming

Eric Walkingshaw
Oregon State University

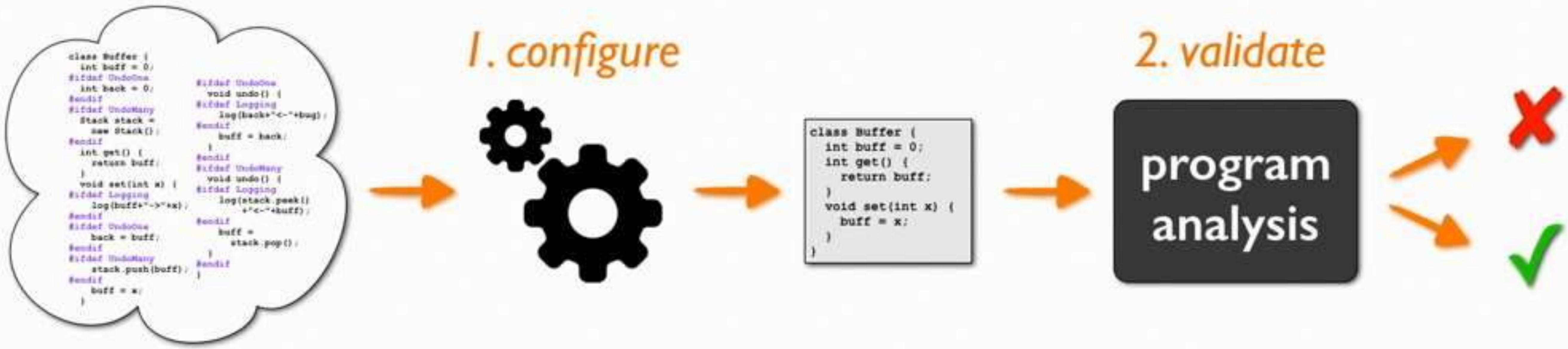
# PL Group @ OSU

# Original motivation: highly configurable software systems

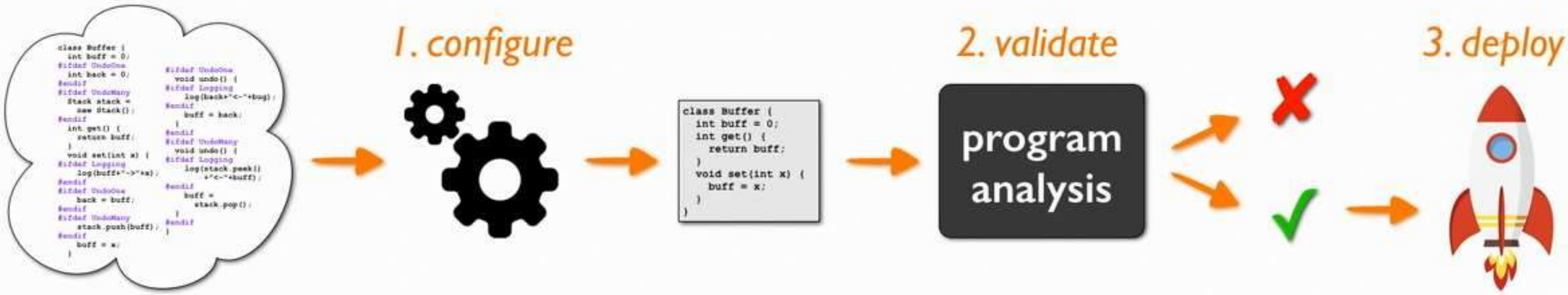
# Original motivation: highly configurable software systems



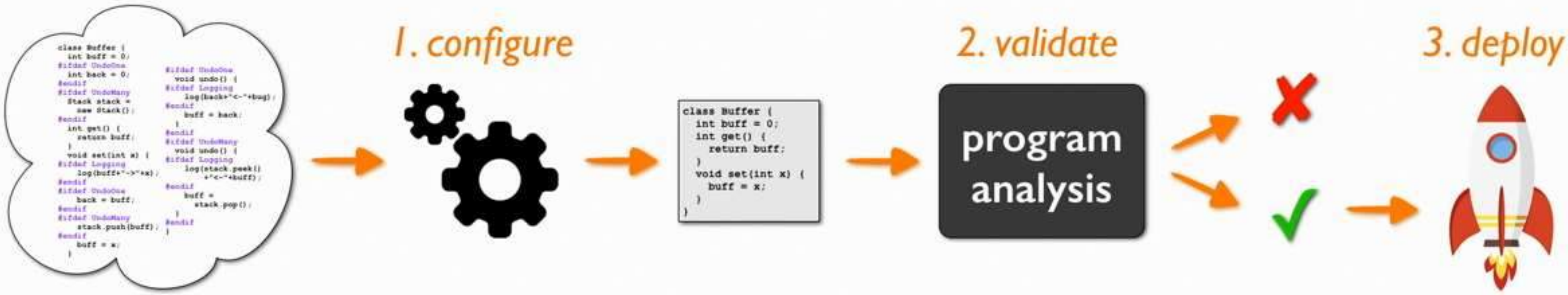
# Original motivation: highly configurable software systems



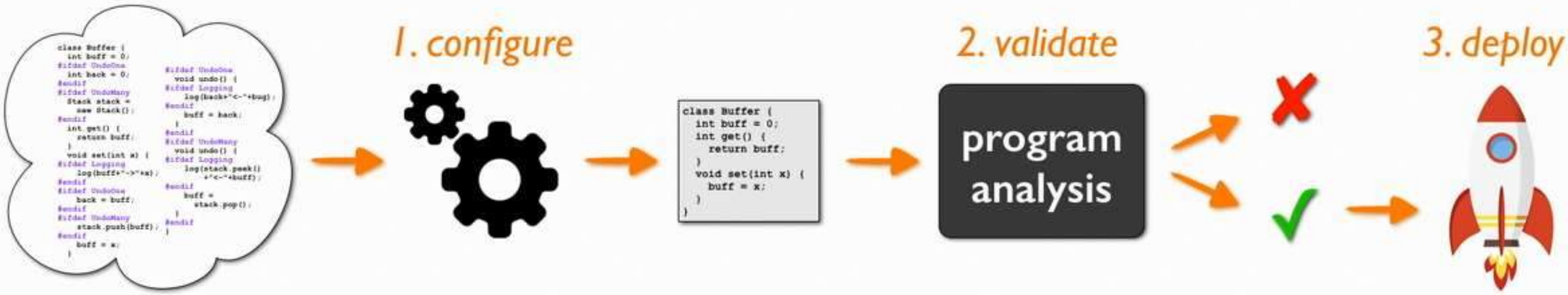
# Original motivation:
# highly configurable software systems



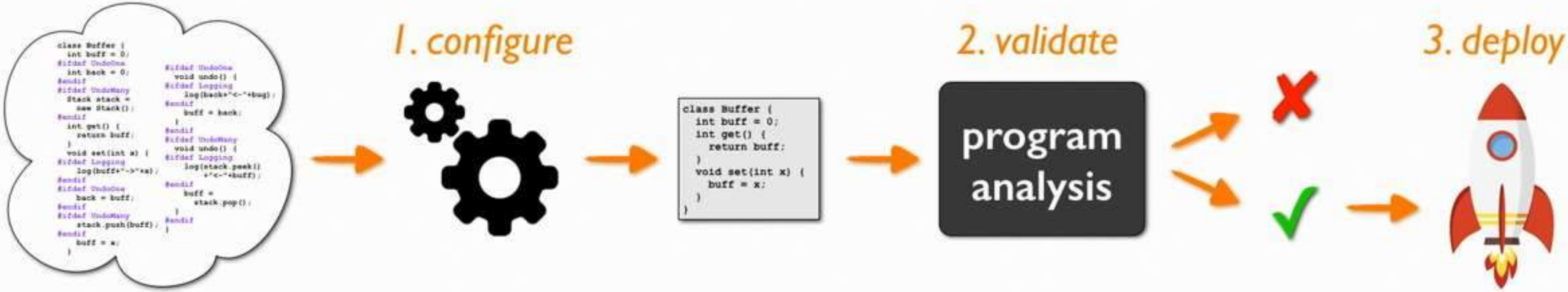
# Original motivation:
# highly configurable software systems




# Original motivation: highly configurable software systems



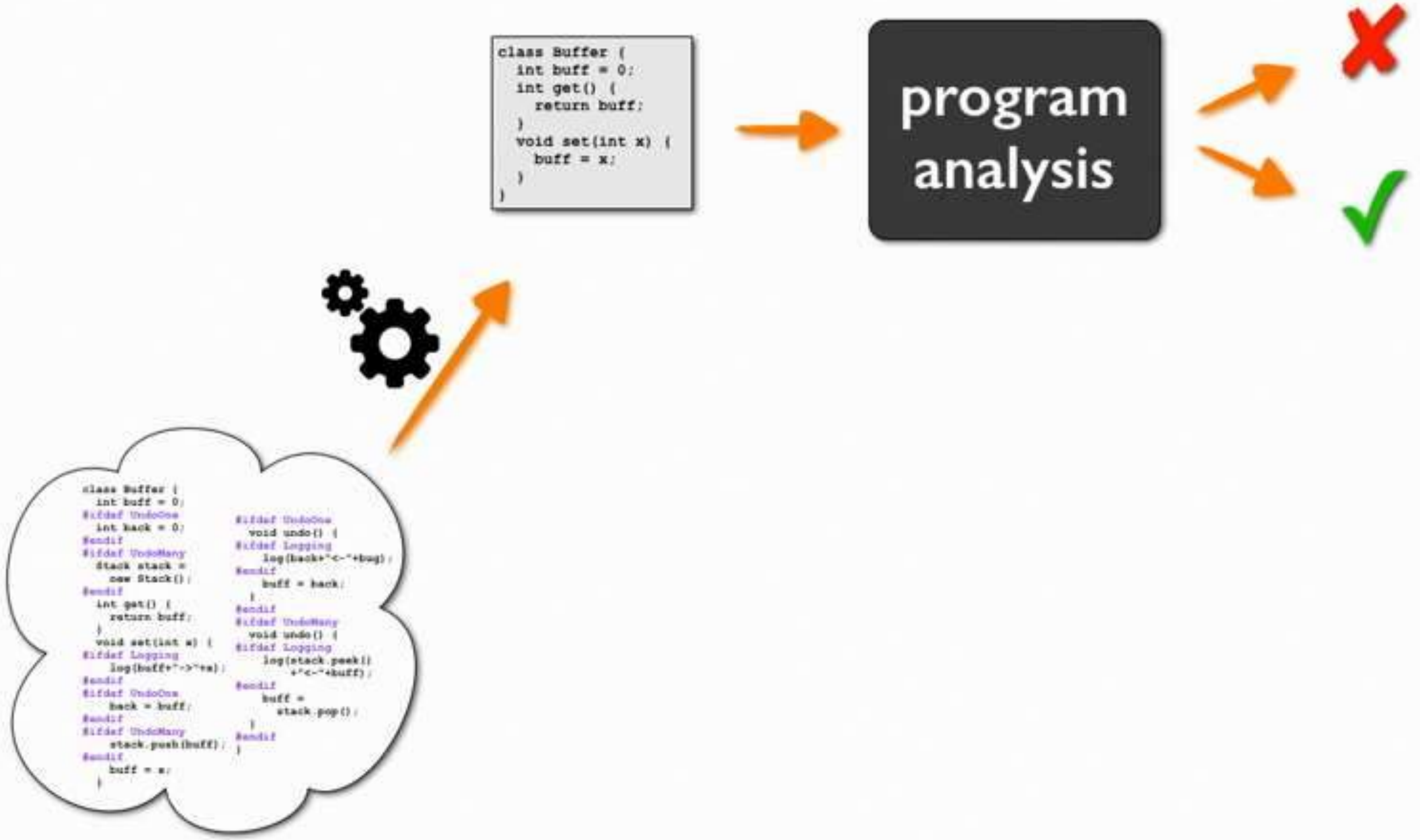

# Original motivation: highly configurable software systems



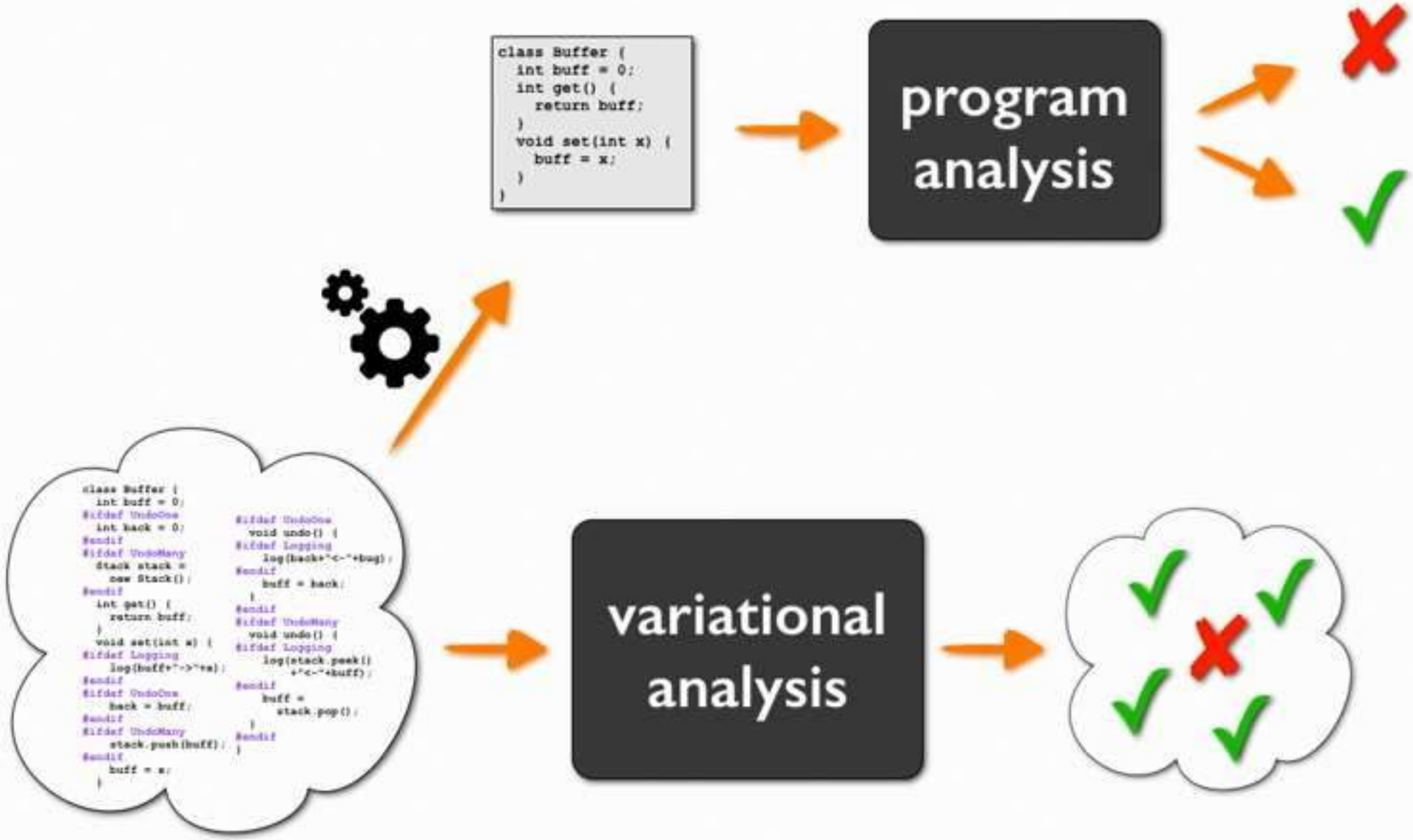

This process finds errors too late!

… but way too many configs to check them all

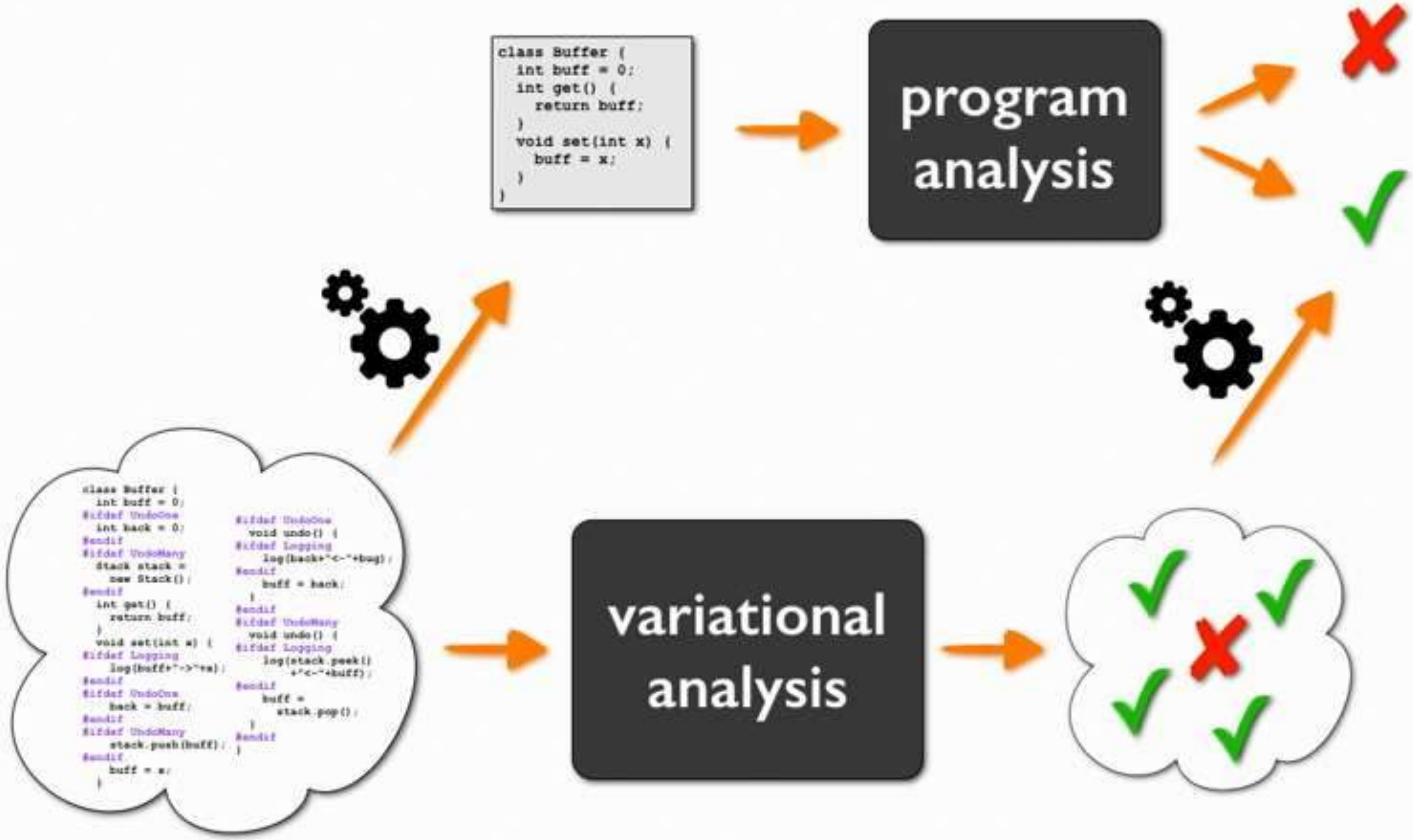
# Solution: variational analyses



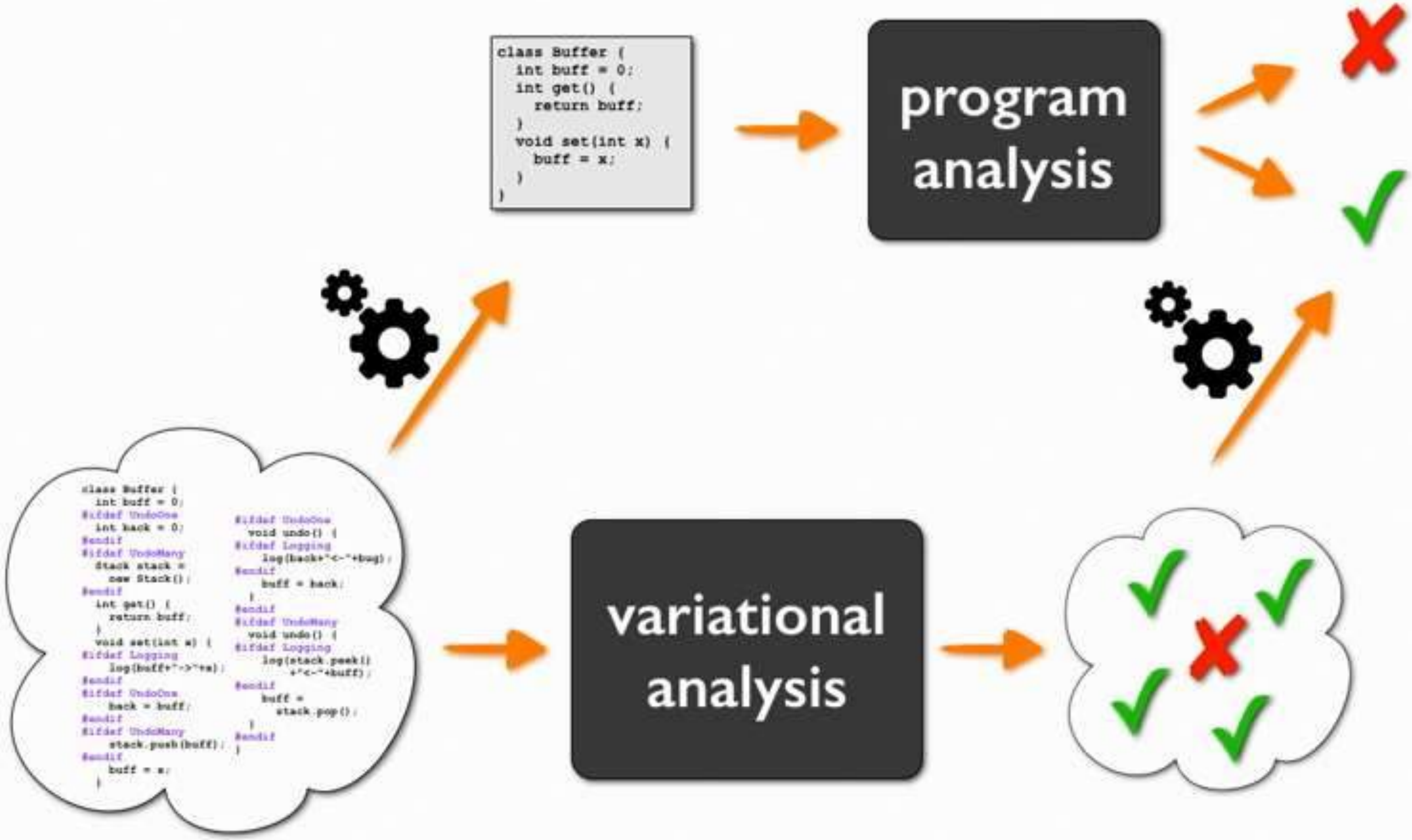
# Solution: variational analyses



# Solution: variational analyses



# Solution: variational analyses

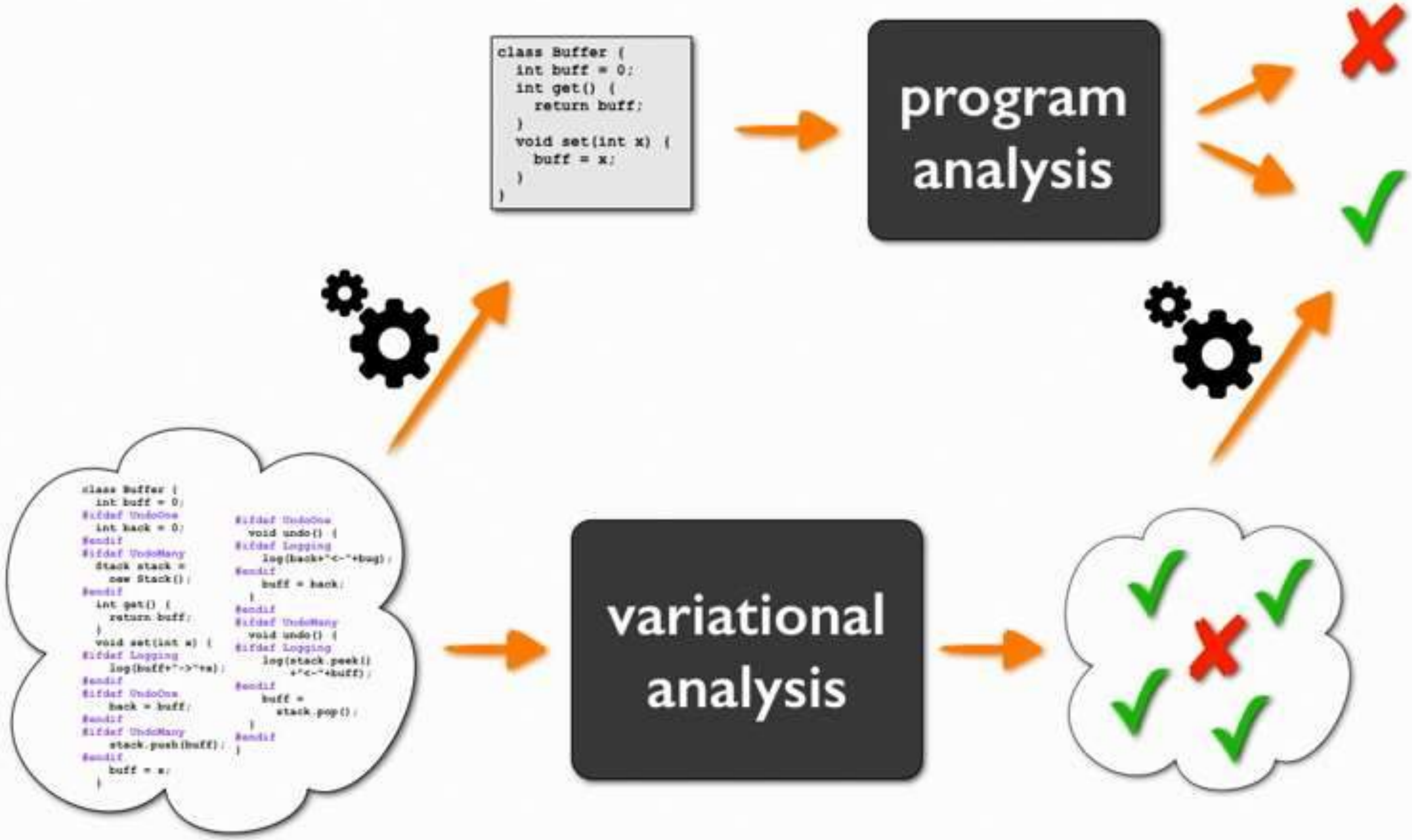



**Our work:**

- languages
- theory
- data structures
- algorithms

to do this *correctly* and *efficiently*

# Solution: variational analyses



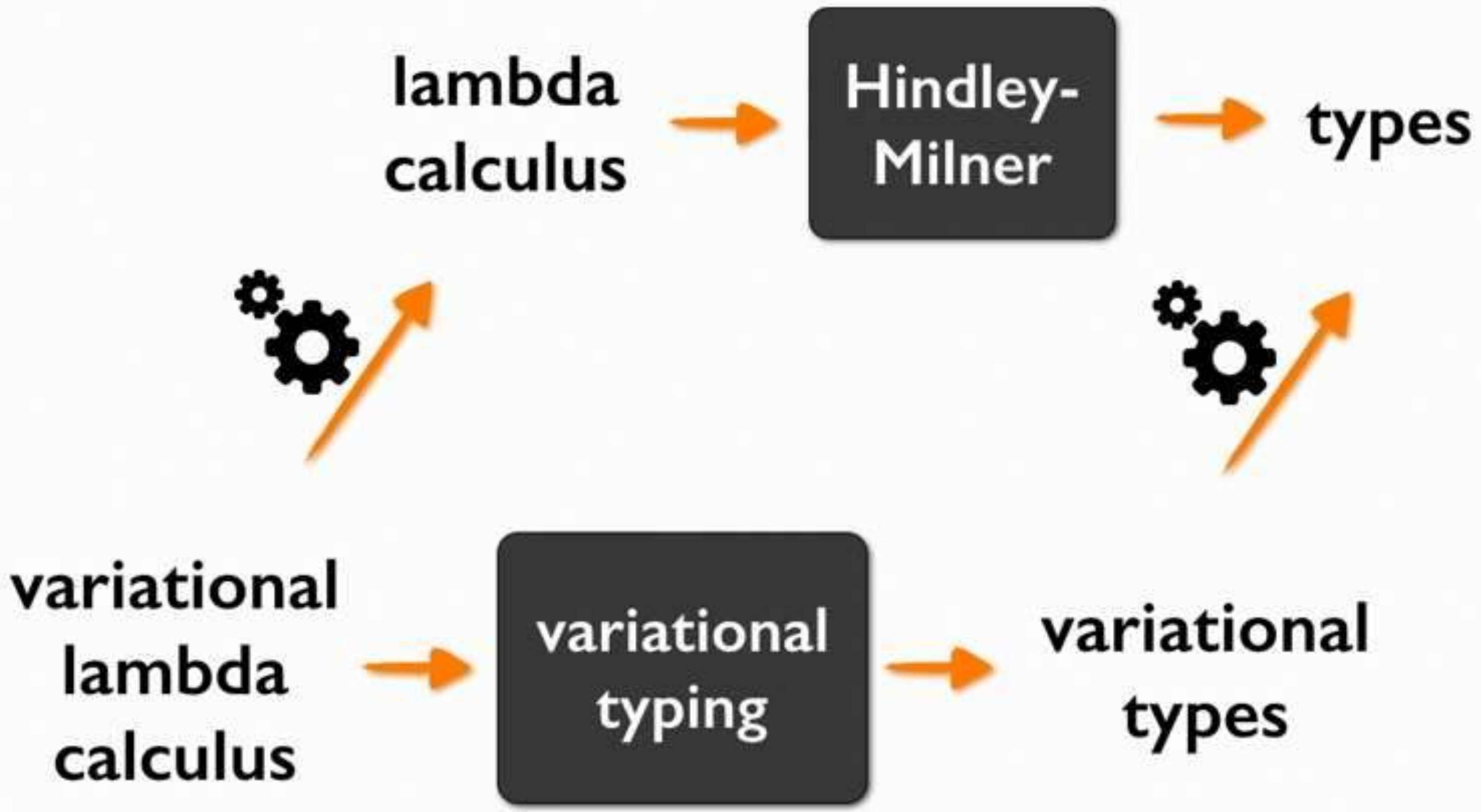

**Our work:**

- languages
- theory
- data structures
- algorithms

to do this *correctly* and *efficiently*

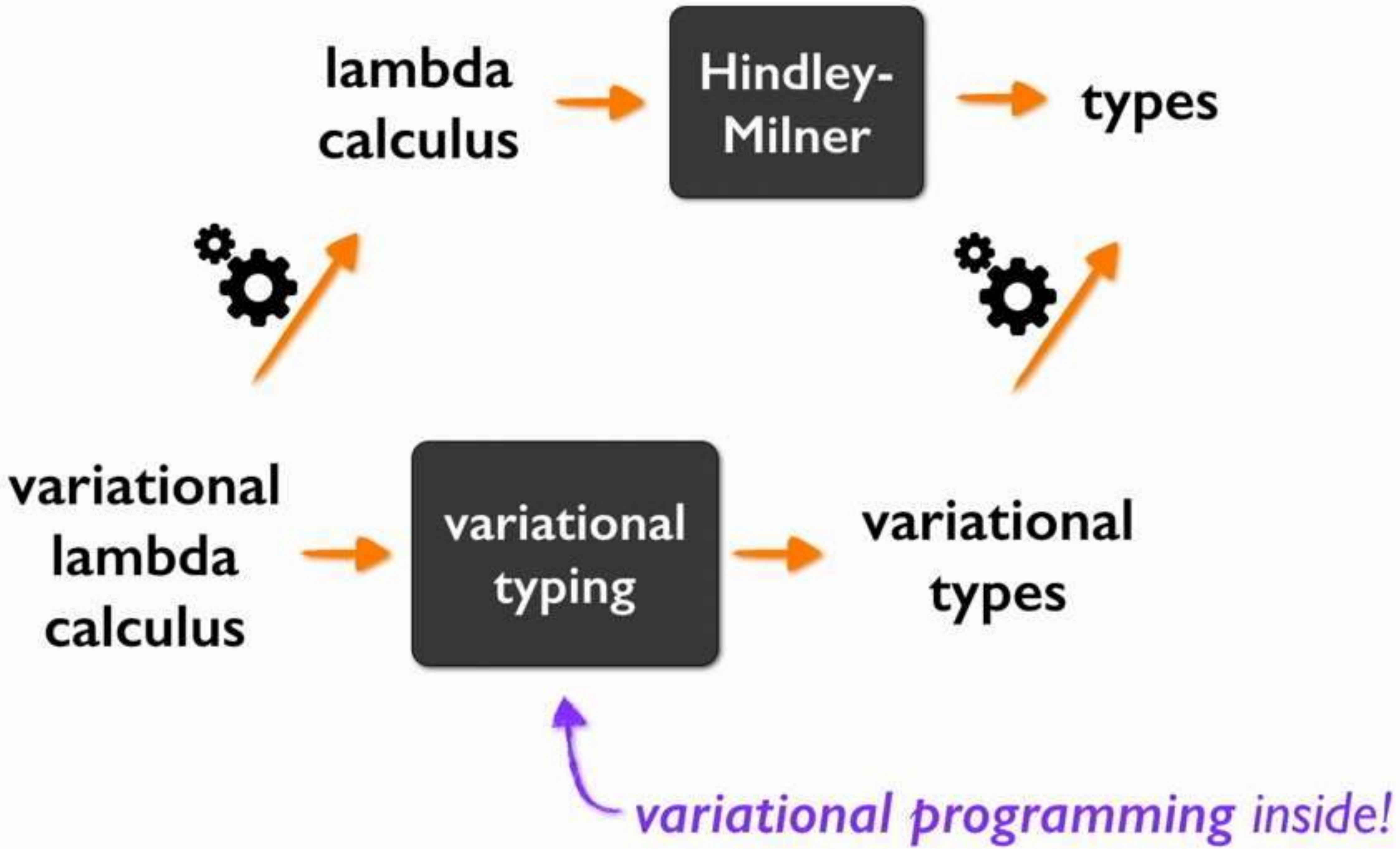
**"variational programming"**

# Example: variational typing




ICFP 2012, TOPLAS 2014

# Example: variational typing




ICFP 2012, TOPLAS 2014

# Variational programming by example

# Variational programming by example

```
A⟨2,3⟩ + 4
```

# Variational programming by example

```
A⟨2,3⟩ + 4
↦ A⟨6,7⟩
```

# Variational programming by example

```
A⟨2,3⟩ + 4
↦ A⟨6,7⟩

A⟨2,3⟩ + A⟨10,20⟩
```

# Variational programming by example

```
A⟨2,3⟩ + 4
↦ A⟨6,7⟩

A⟨2,3⟩ + A⟨10,20⟩
↦ A⟨12,23⟩
```

# Variational programming by example

```
A⟨2,3⟩ + 4
↦ A⟨6,7⟩

A⟨2,3⟩ + A⟨10,20⟩
↦ A⟨12,23⟩

A⟨2,3⟩ + B⟨10,20⟩
```

# Variational programming by example

```
A⟨2,3⟩ + 4
↦ A⟨6,7⟩
```

```
A⟨2,3⟩ + A⟨10,20⟩
↦ A⟨12,23⟩
```

```
A⟨2,3⟩ + B⟨10,20⟩
↦ A⟨B⟨12,22⟩,B⟨13,23⟩⟩
```

# Variational programming by example

```
A⟨2,3⟩ + 4
↦ A⟨6,7⟩

A⟨2,3⟩ + A⟨10,20⟩
↦ A⟨12,23⟩

A⟨2,3⟩ + B⟨10,20⟩
↦ A⟨B⟨12,22⟩,B⟨13,23⟩⟩
```

```
A⟨True,3⟩
```

# Variational programming by example

```
A⟨2,3⟩ + 4
↦ A⟨6,7⟩

A⟨2,3⟩ + A⟨10,20⟩
↦ A⟨12,23⟩

A⟨2,3⟩ + B⟨10,20⟩
↦ A⟨B⟨12,22⟩,B⟨13,23⟩⟩
```

```
A⟨True,3⟩
: A⟨Bool,Int⟩
```

# Variational programming by example

```
A⟨2,3⟩ + 4
↦ A⟨6,7⟩

A⟨2,3⟩ + A⟨10,20⟩
↦ A⟨12,23⟩

A⟨2,3⟩ + B⟨10,20⟩
↦ A⟨B⟨12,22⟩,B⟨13,23⟩⟩
```

```
A⟨True,3⟩
: A⟨Bool,Int⟩

A⟨succ,even⟩
```

# Variational programming by example

```
A⟨2,3⟩ + 4
↦ A⟨6,7⟩
```

```
A⟨2,3⟩ + A⟨10,20⟩
↦ A⟨12,23⟩
```

```
A⟨2,3⟩ + B⟨10,20⟩
↦ A⟨B⟨12,22⟩,B⟨13,23⟩⟩
```

```
A⟨True,3⟩
: A⟨Bool,Int⟩
```

```
A⟨succ,even⟩
: Int -> A⟨Int,Bool⟩
```

# Variational programming by example

```
A⟨2,3⟩ + 4
↦ A⟨6,7⟩

A⟨2,3⟩ + A⟨10,20⟩
↦ A⟨12,23⟩

A⟨2,3⟩ + B⟨10,20⟩
↦ A⟨B⟨12,22⟩,B⟨13,23⟩⟩
```

```
A⟨True,3⟩
: A⟨Bool,Int⟩

A⟨succ,even⟩
: Int -> A⟨Int,Bool⟩
≡ A⟨Int -> Int,Int -> Bool⟩
```

# Variational programming by example

```
A⟨not,even⟩ A⟨True,3⟩
```

# Variational programming by example

```
A⟨not,even⟩ A⟨True,3⟩
↦ False
```

# Variational programming by example

```
A⟨not,even⟩ A⟨True,3⟩
↦ False
≡ A⟨False,False⟩
```

# Variational programming by example

```
A⟨not,even⟩ A⟨True,3⟩
↦ False
≡ A⟨False,False⟩

A⟨succ,even⟩ B⟨2,4⟩
```

# Variational programming by example

```
A⟨not,even⟩ A⟨True,3⟩
↦ False
≡ A⟨False,False⟩

A⟨succ,even⟩ B⟨2,4⟩
↦ A⟨B⟨3,5⟩,True⟩
```

# Variational programming by example

```
A⟨not,even⟩ A⟨True,3⟩
↦ False
≡ A⟨False,False⟩
```

```
A⟨succ,even⟩ B⟨2,4⟩
↦ A⟨B⟨3,5⟩,True⟩
```

```
vsum (A⟨2,3⟩ + B⟨10,20⟩)
```

# Variational programming by example

```
A⟨not,even⟩ A⟨True,3⟩
↦ False
≡ A⟨False,False⟩

A⟨succ,even⟩ B⟨2,4⟩
↦ A⟨B⟨3,5⟩,True⟩
```

```
vsum (A⟨2,3⟩ + B⟨10,20⟩)
↦ 70
```

# Variational programming by example

```
A⟨not,even⟩ A⟨True,3⟩
↦ False
≡ A⟨False,False⟩

A⟨succ,even⟩ B⟨2,4⟩
↦ A⟨B⟨3,5⟩,True⟩
```

```
vsum (A⟨2,3⟩ + B⟨10,20⟩)
↦ 70

vmax (A⟨2,3⟩ + B⟨10,20⟩)
```

# Variational programming by example

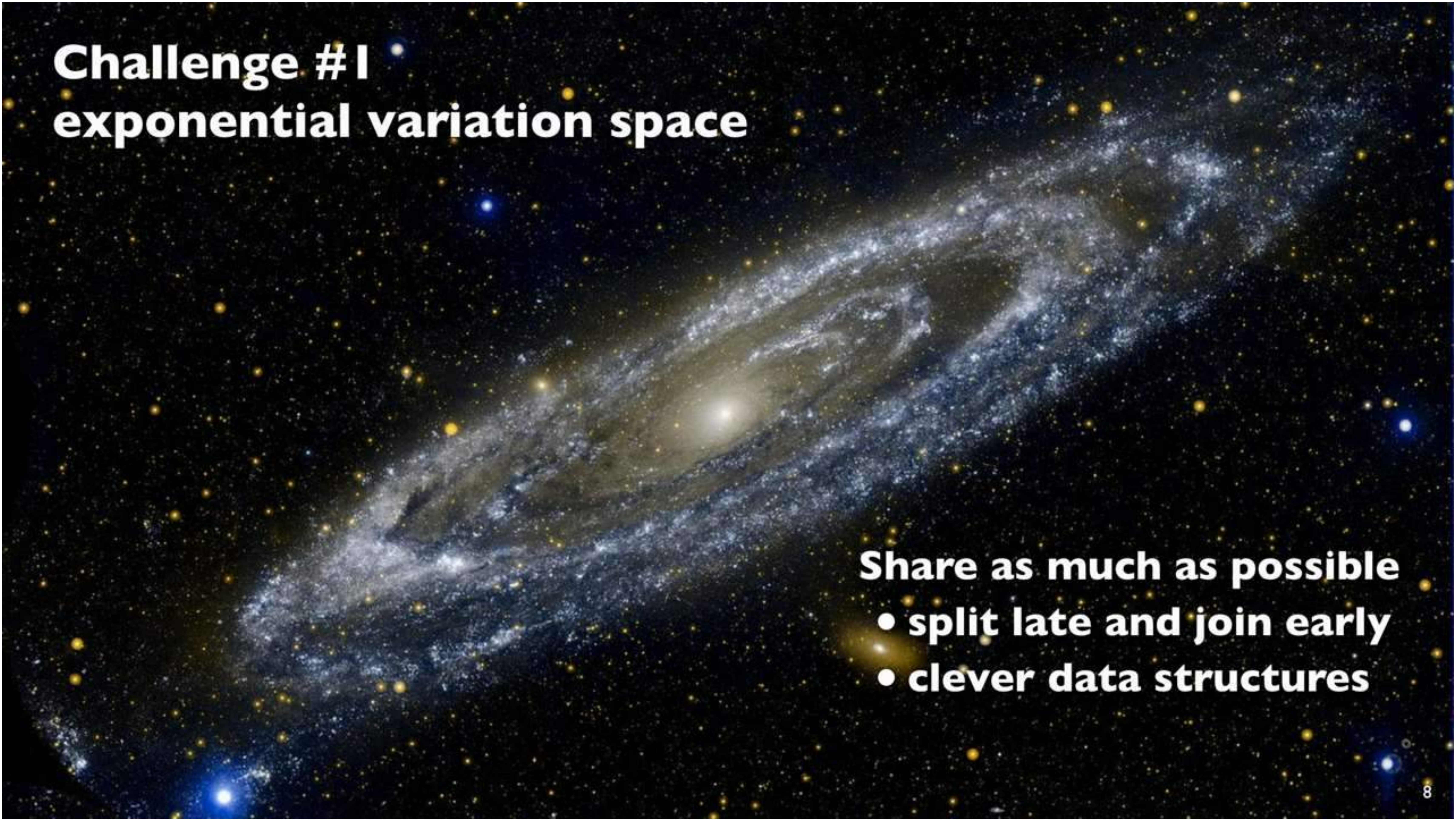
```
A⟨not,even⟩ A⟨True,3⟩
↦ False
≡ A⟨False,False⟩
```

```
vsum (A⟨2,3⟩ + B⟨10,20⟩)
↦ 70
```

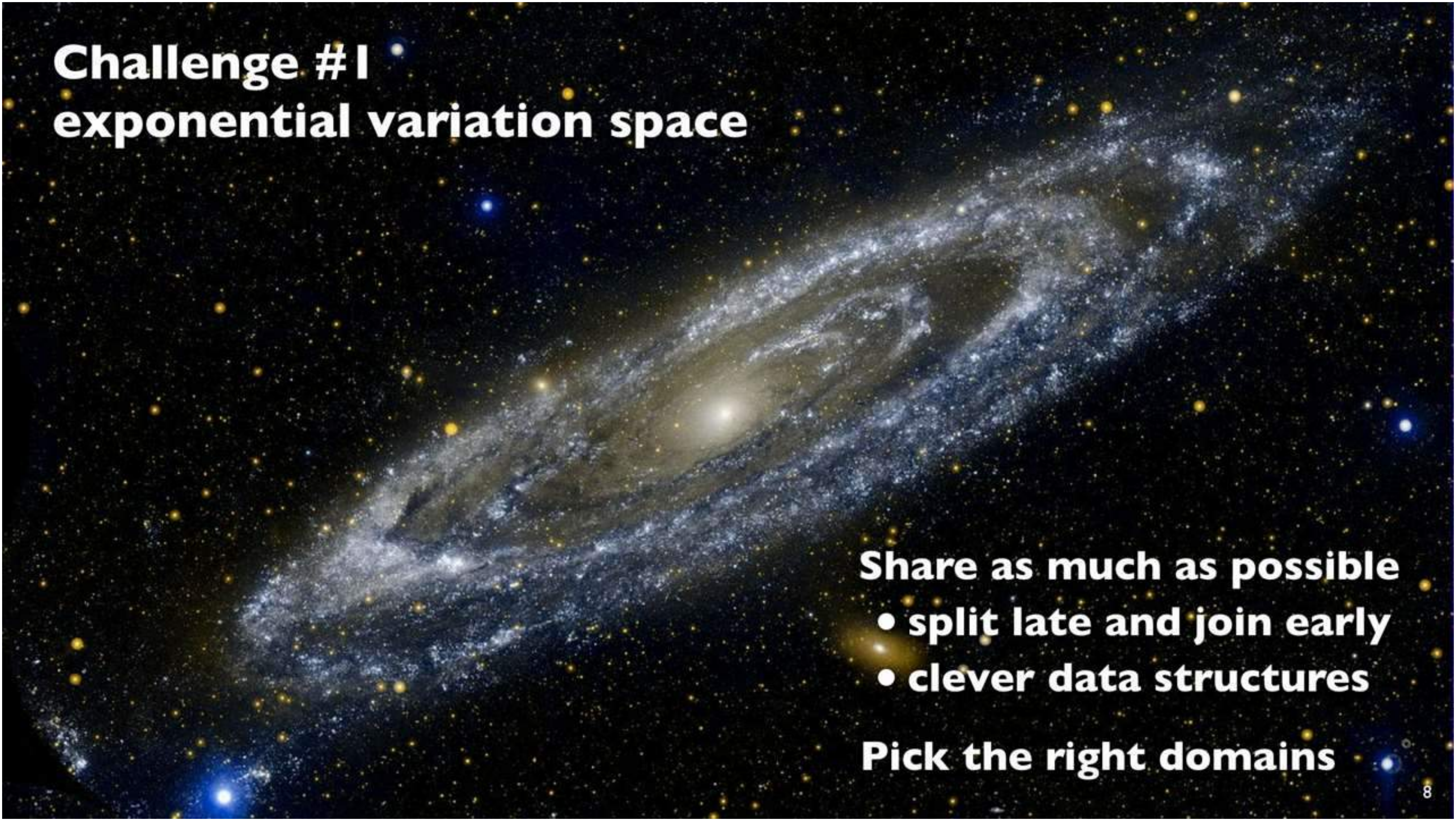
```
A⟨succ,even⟩ B⟨2,4⟩
↦ A⟨B⟨3,5⟩,True⟩
```

```
vmax (A⟨2,3⟩ + B⟨10,20⟩)
↦ 23 @ [A.R,B.R]
```

# Challenge #1
# exponential variation space


Challenge #1
exponential variation space
Share as much as possible
• split late and join early
• clever data structures

# Challenge #1
# exponential variation space

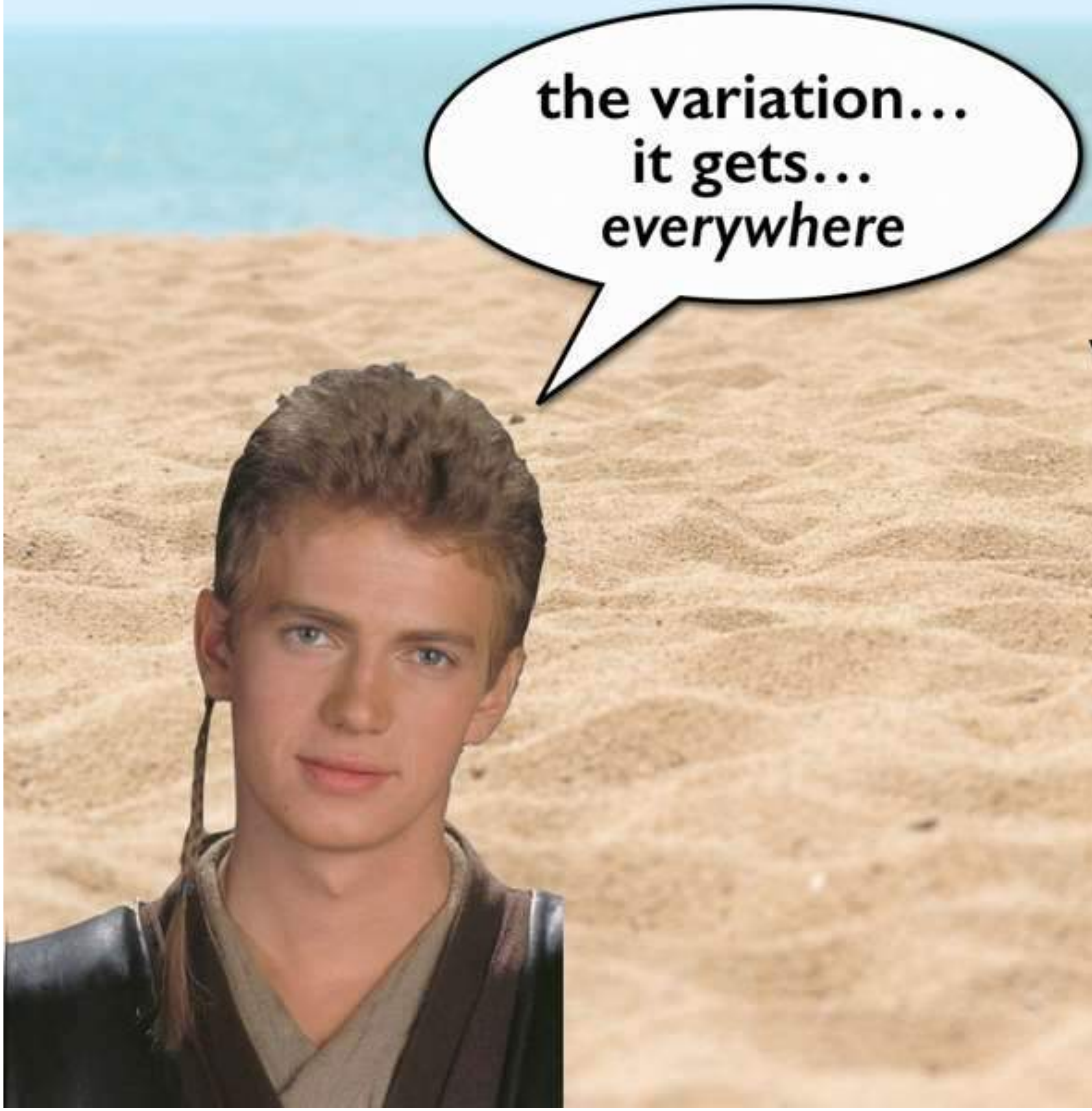


**Share as much as possible**
- split late and join early
- clever data structures

**Pick the right domains**

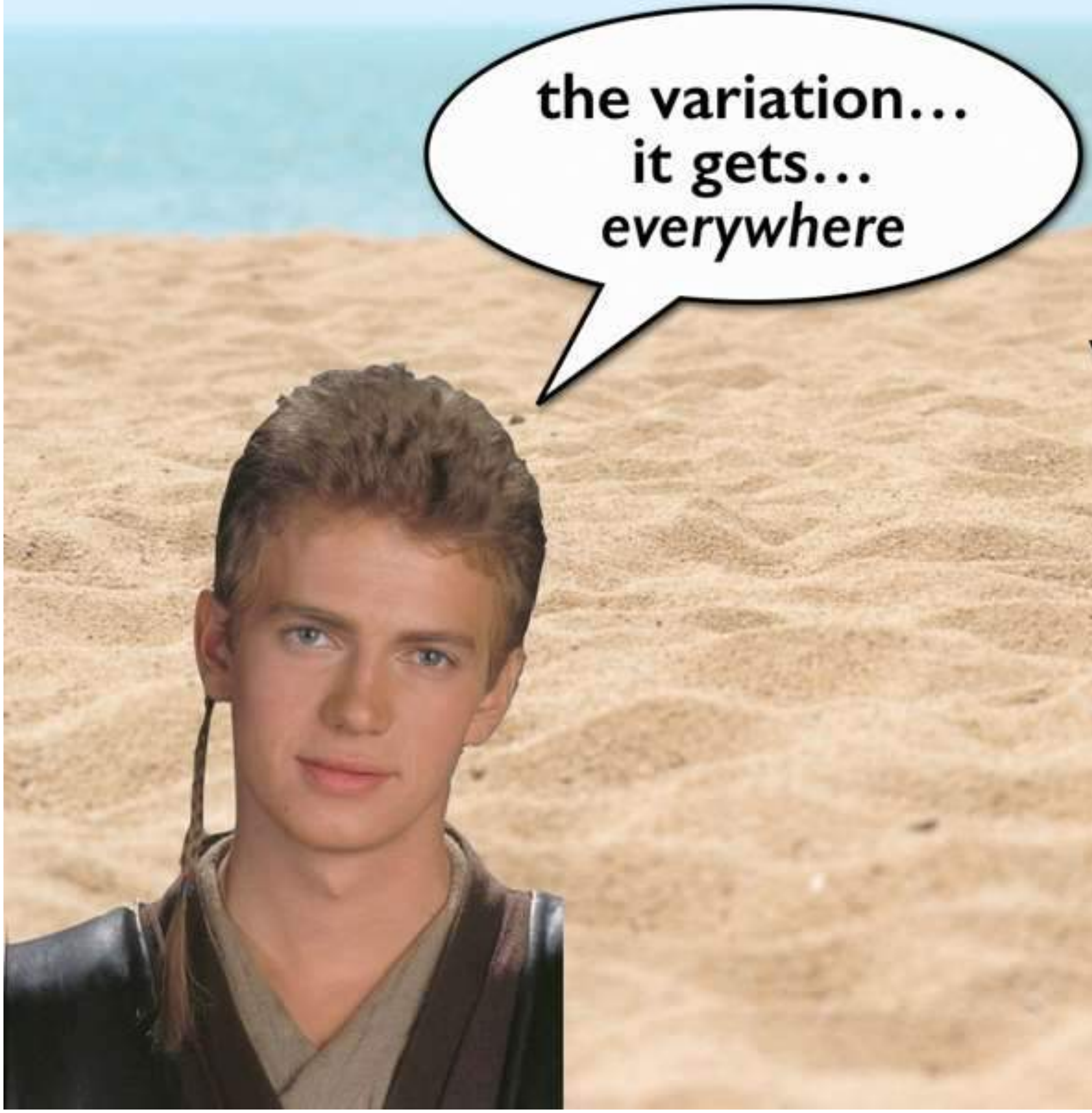
# Challenge #2

# Challenge #2




**Variation in structured data**

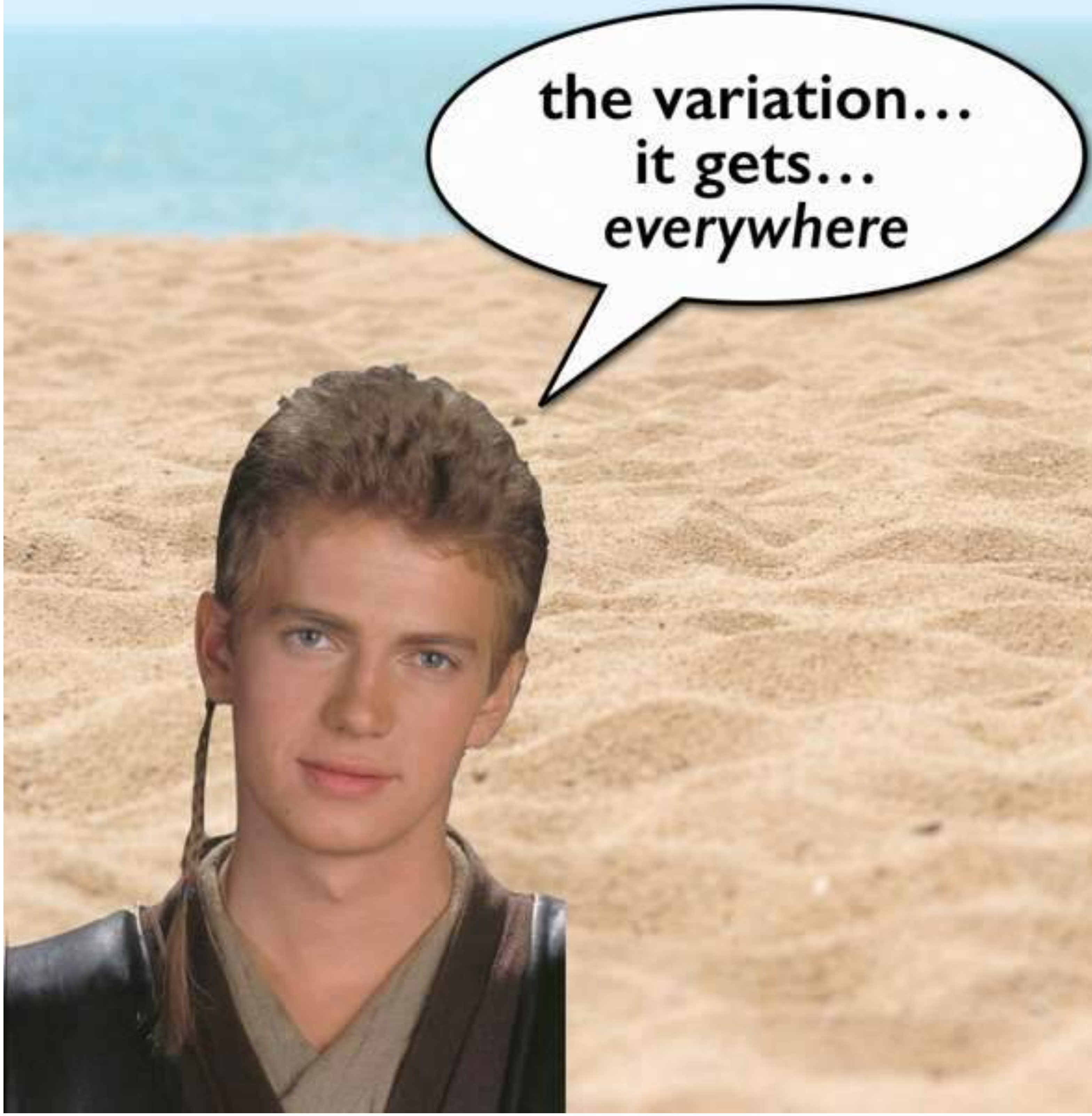
# Challenge #2




**Variation in structured data**

```
A⟨[1,2,3],[1,2,4,5]⟩
```

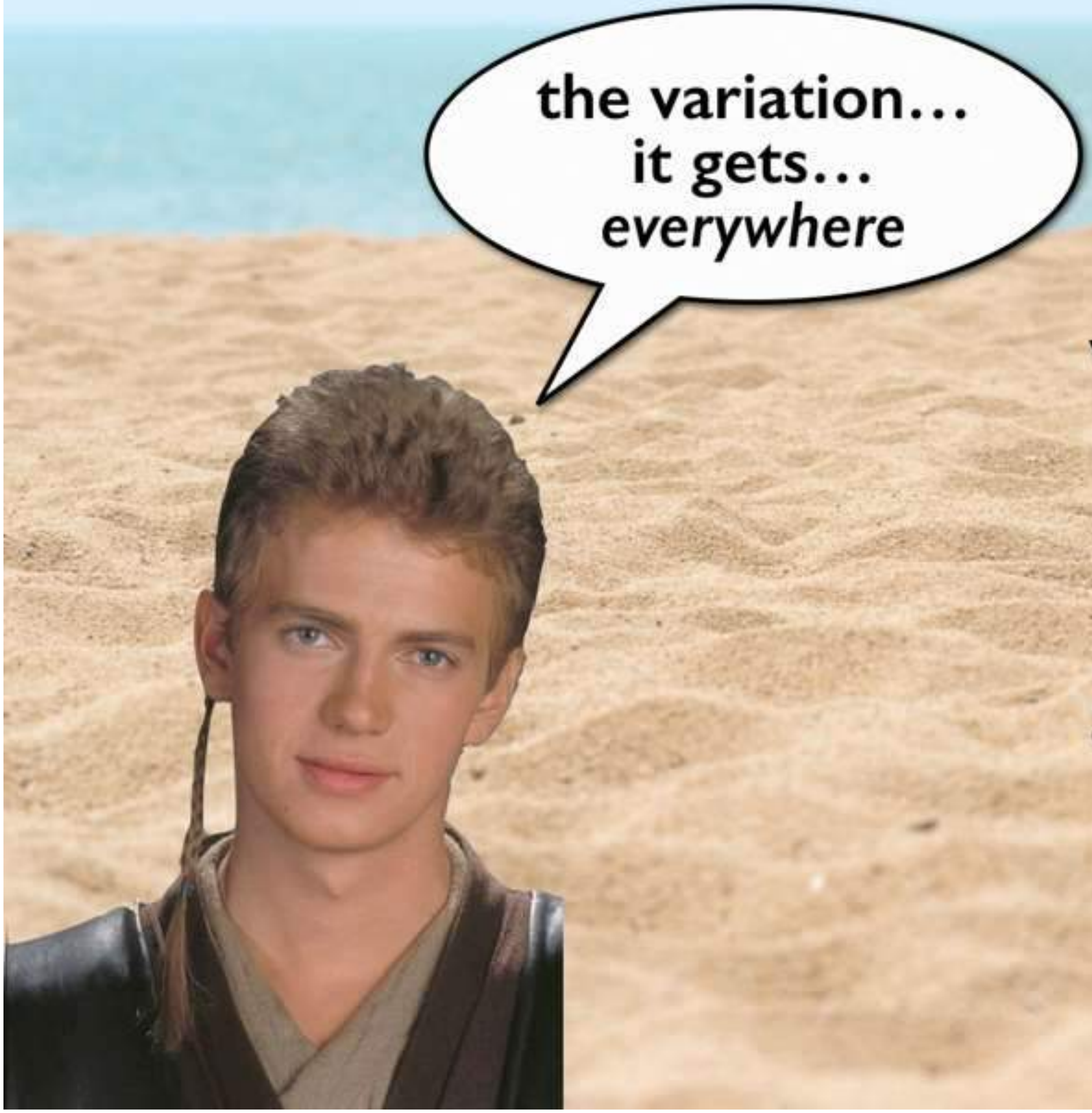
# Challenge #2




**Variation in structured data**

```
A⟨[1,2,3],[1,2,4,5]⟩
≡ [1,2,A⟨3,4⟩,A⟨5,_⟩] ??
```

# Challenge #2

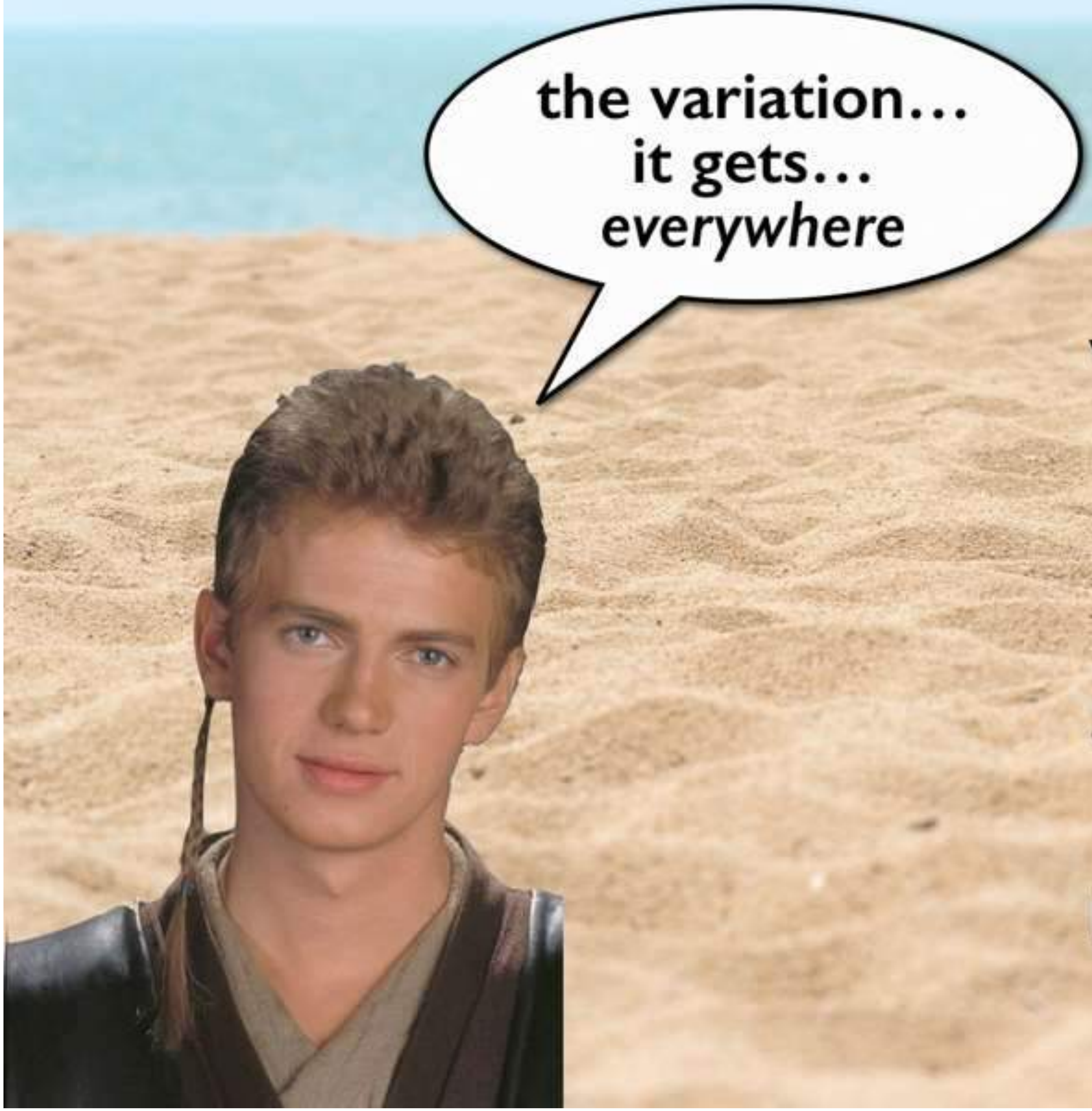



**Variation in structured data**

```
A⟨[1,2,3],[1,2,4,5]⟩
≡ [1,2,A⟨3,4⟩,A⟨5,_⟩] ??
```

**… need *variational data structures***

# Challenge #2



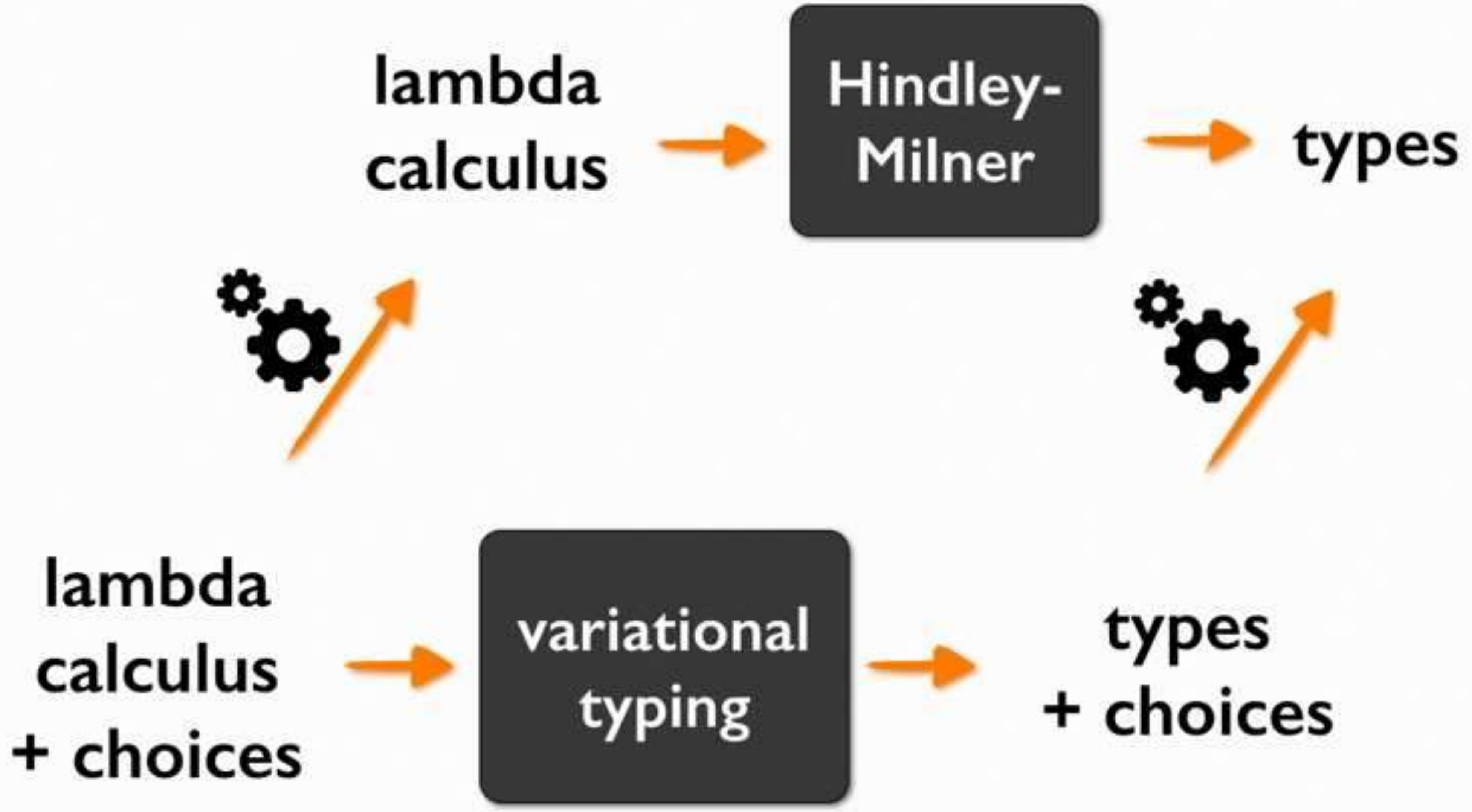

**Variation in structured data**

```
A⟨[1,2,3],[1,2,4,5]⟩
≡ [1,2,A⟨3,4⟩,A⟨5,_⟩] ??
```

**… need *variational data structures***

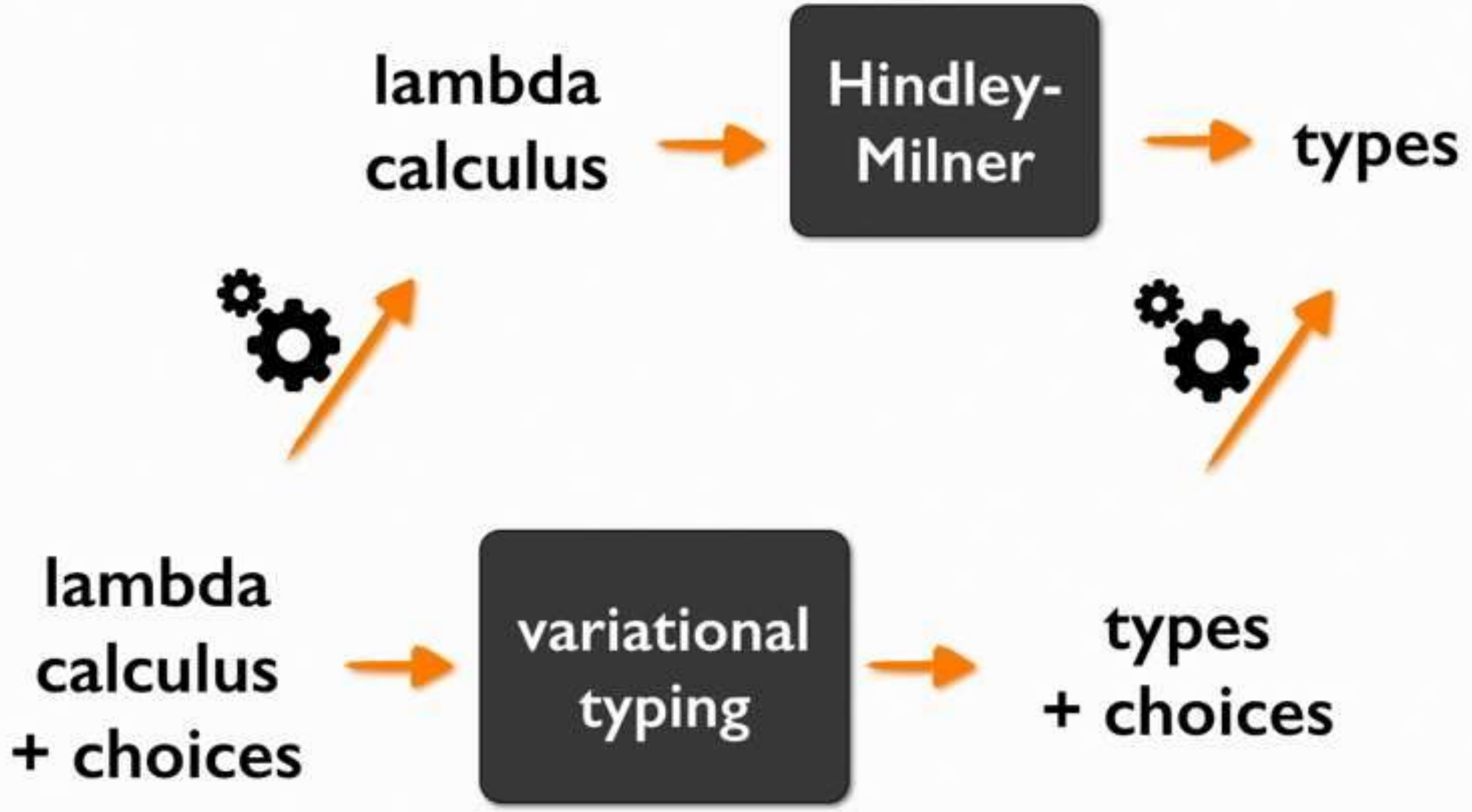
**How to handle side effects?**

# Variational typing



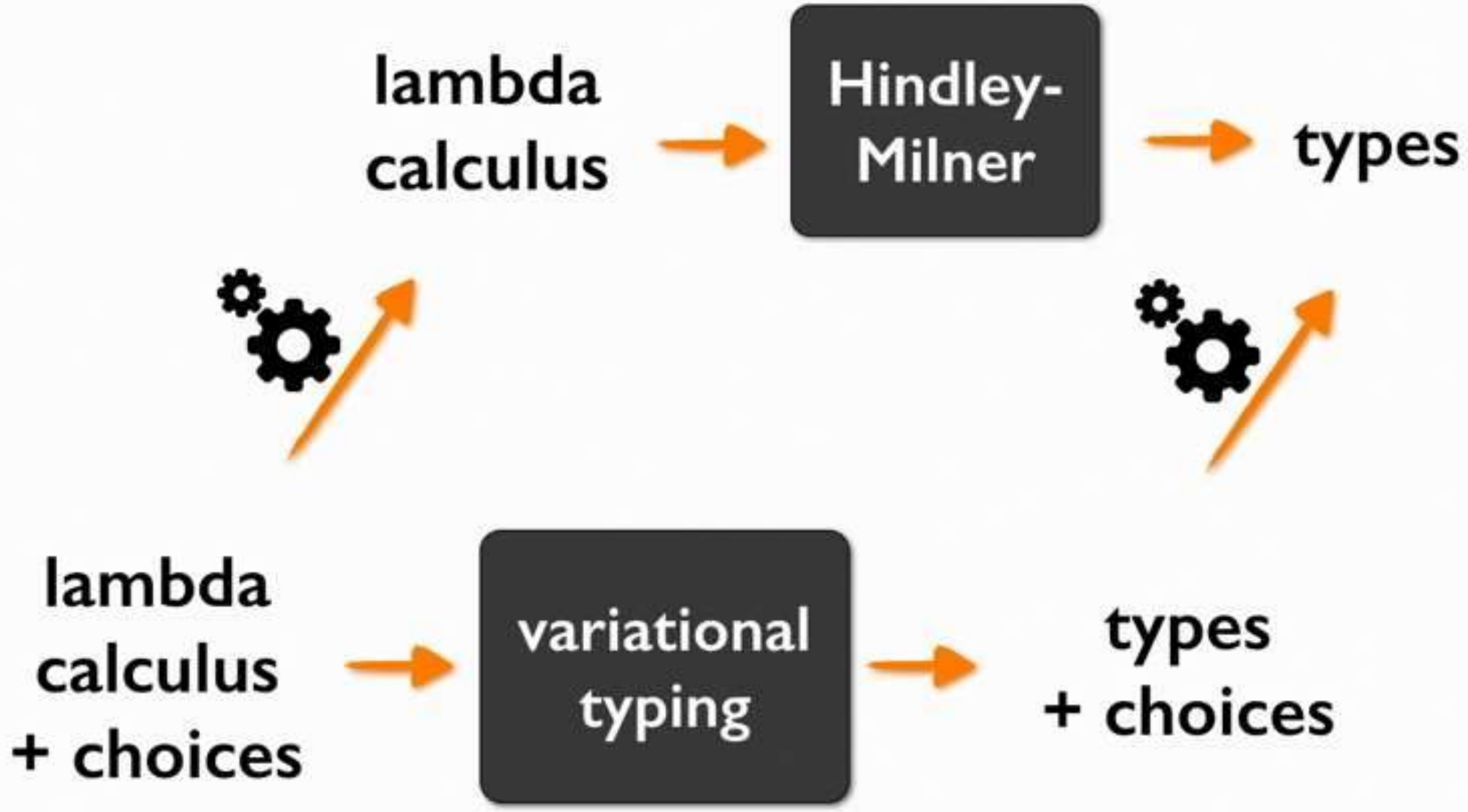

ICFP 2012, TOPLAS 2014

# Variational typing




*Lots of sharing*

# Variational typing

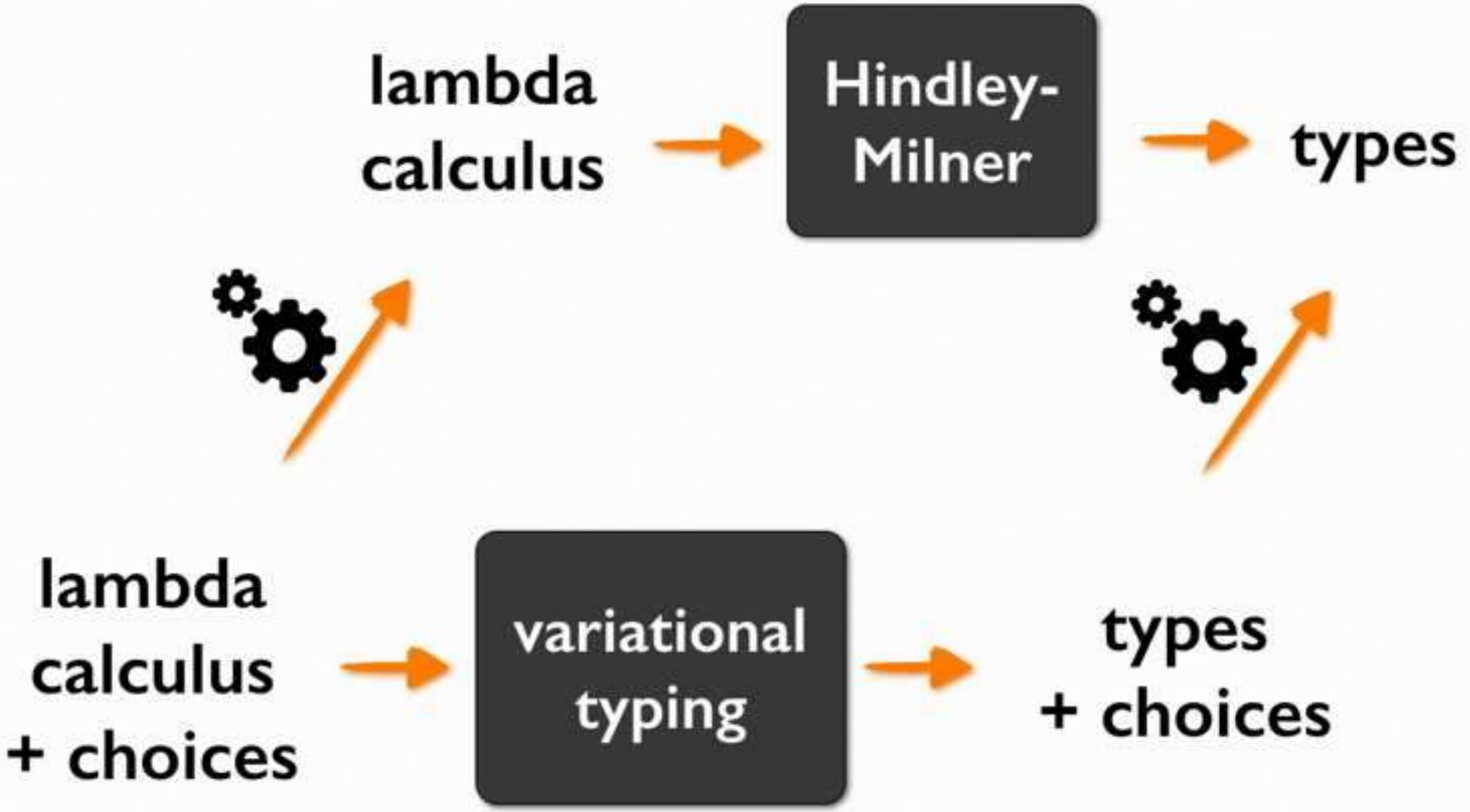



*Lots* of sharing

Scales roughly linearly w/ size of program

ICFP 2012, TOPLAS 2014

# Variational typing




*Lots* of sharing

Scales roughly linearly w/ size of program

Scales to Linux kernel (Kästner et al.)

# Only good for product line analyses?

# Only good for product line analyses?

**What makes a good nail?**

- lots of variation
- lots of sharing

# Only good for product line analyses?

**What makes a good nail?**

- lots of variation
- lots of sharing

**Idea: use variation to explore hypothetical scenarios …**

# Error location

```
fold f z []     = [z]
fold f z (h:t) = fold f (f z h) t

flip f x y = f y x

reverse = fold (flip (:)) []

palindrome xs = reverse xs == xs
```

# Error location

```
fold f z []    = [z]
fold f z (h:t) = fold f (f z h) t

flip f x y = f y x

reverse = fold (flip (:)) []

palindrome xs = reverse xs == xs
```

```
• Occurs check: cannot construct the infinite type: a ~ [a]
  Expected type: [[a]]
  Actual type: [a]
• In the second argument of ‘(==)’, namely ‘xs’
  In the expression: reverse xs == xs
```

# Error location

```
fold f z []     = [z]
fold f z (h:t) = fold f (f z h) t

flip f x y = f y x

reverse = fold (flip (:)) []

palindrome xs = reverse xs == xs
```

*error is in base case of fold*

*… but fold type checks!*

*use of fold also type checks!*

*error finally detected*

```
• Occurs check: cannot construct the infinite type: a ~ [a]
  Expected type: [[a]]
  Actual type: [a]
• In the second argument of '(==)', namely 'xs'
  In the expression: reverse xs == xs
```

# Counterfactual typing

**Problem:** locating the cause of a type error is hard

- type inference *commits* too early
- a successfully inferred type could be wrong!

Chen and Erwig
POPL 2014, JFP 2018

# Error location

```
fold f z []     = [z]
fold f z (h:t) = fold f (f z h) t

flip f x y = f y x

reverse = fold (flip (:)) []

palindrome xs = reverse xs == xs
```

*error is in base case of fold*
*… but fold type checks!*

*use of fold also type checks!*

*error finally detected*

```
• Occurs check: cannot construct the infinite type: a ~ [a]
  Expected type: [[a]]
  Actual type: [a]
• In the second argument of ‘(==)’, namely ‘xs’
  In the expression: reverse xs == xs
```

# Counterfactual typing

**Problem:** locating the cause of a type error is hard

- type inference *commits* too early
- a successfully inferred type could be wrong!

Chen and Erwig
POPL 2014, JFP 2018

# Counterfactual typing

**Problem:** locating the cause of a type error is hard

- type inference *commits* too early
- a successfully inferred type could be wrong!

**Solution:**


Chen and Erwig
POPL 2014, JFP 2018

# Counterfactual typing

**Problem:** locating the cause of a type error is hard

- type inference *commits* too early
- a successfully inferred type could be wrong!

**Solution:**


1. Error-tolerant variational type inference where for *every* subexpression

`e : T` → `e : d⟨T,a⟩`     `d` & `a` *are fresh*

Chen and Erwig
POPL 2014, JFP 2018

# Counterfactual typing

**Problem:** locating the cause of a type error is hard

- type inference *commits* too early
- a successfully inferred type could be wrong!

**Solution:**


1. Error-tolerant variational type inference where for *every* subexpression

   `e : T` → `e : d⟨T,a⟩`   `d` & `a` *are fresh*

2. Search output variational type for
   - non-error type
   - as few right selections as possible

Chen and Erwig
POPL 2014, JFP 2018

# Migrating gradual types

```
def f(mode:bool, x):
  if mode:
    return even(x)
  else:
    return not(x)
```

**Gradual typing**

mix static and dynamic types in the same program

# Migrating gradual types

```
def f(mode:bool, x):
  if mode:
    return even(x)
  else:
    return not(x)
```

**Gradual typing**
mix static and dynamic types
in the same program

Migration challenges: (adding/removing annotations)

- mutually exclusive annotations

# Migrating gradual types

```
def f(mode:bool, x):
  if mode:
    return even(x)
  else:
    return not(x)
```

**Gradual typing**
mix static and dynamic types
in the same program

Migration challenges: (adding/removing annotations)

- mutually exclusive annotations
- local type-safety maxima
- potential for extreme performance degradation

# Migrational typing

**Problem:** migrating gradual types is perilous
and exploration by trial-and-error is infeasible

**Solution:**


1. Every parameter is initially d⟨a,?⟩
   d & a *are fresh* ? *is the dynamic type*

2. Variational gradual type inference + cost analysis
   output = summary of *all possible* migrations

3. Filter/search variational output
   most static = fewest right selections
   cheapest = lowest cost

POPL 2018, ICFP 2018 🤞

# Variational programming take aways

**Key ideas**

- make variation *explicit* in programs and data, then compute with it!
- share as much as possible between variants

**Can efficiently explore lots of hypothetical scenarios**

CRAY

# Chapel Comes of Age: Productive Parallelism at Scale

Brad Chamberlain, Chapel Team, Cray Inc.

PNW PLSE Workshop

May 14, 2018

CRAY CHAPEL

PNW PLSE

# Chapel: Niche or Quiche?

**Brad Chamberlain, Chapel Team, Cray Inc.**

**PNW PLSE Workshop**

**May 14, 2018**

# What is Chapel?

**Chapel:** A productive parallel programming language

- portable & scalable
- open-source & collaborative

# What is Chapel?

**Chapel:** A productive parallel programming language

- portable & scalable
- open-source & collaborative

**Goals:**

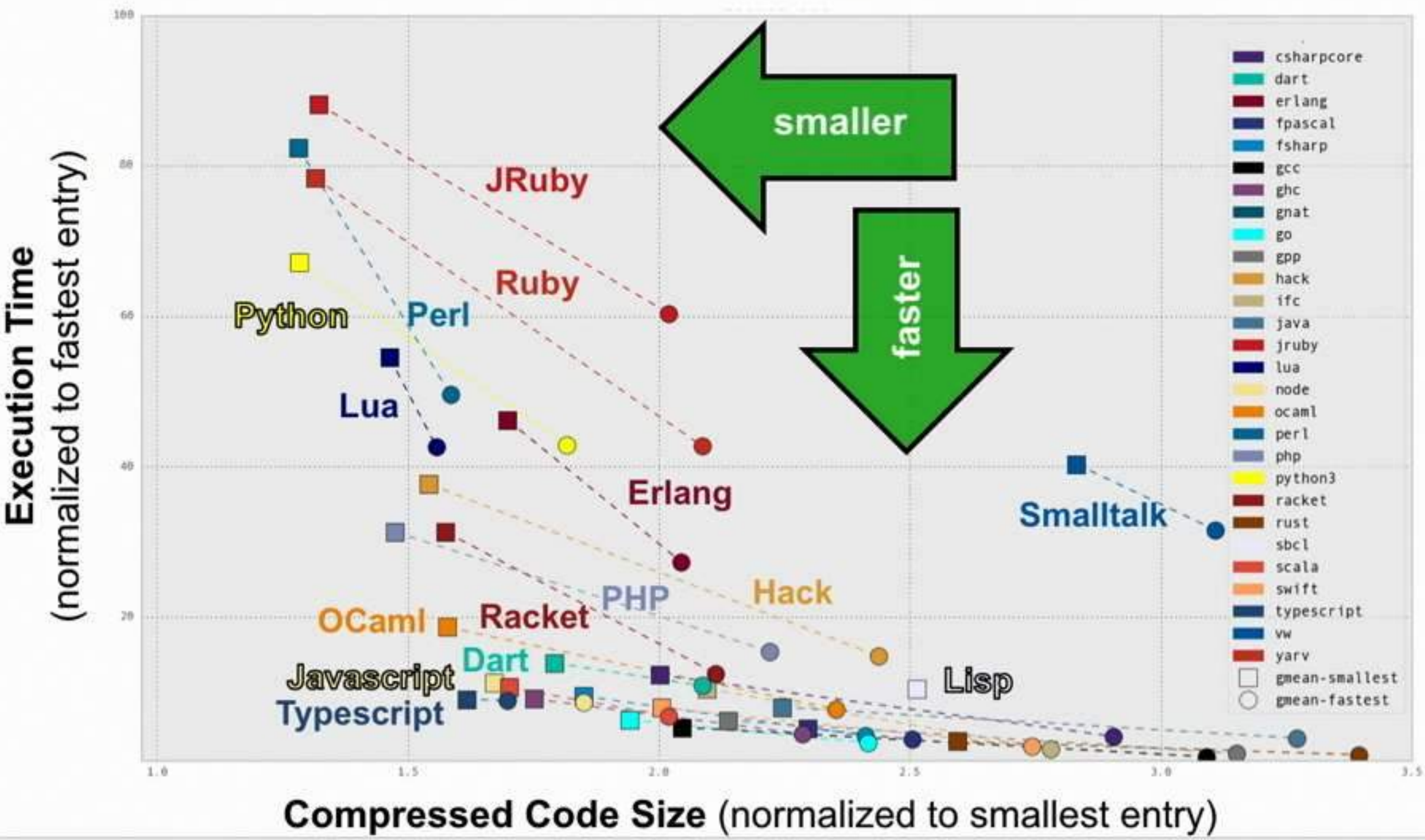
- Support general parallel programming
  - "any parallel algorithm on any parallel hardware"
- Make parallel programming at scale far more productive

# Chapel and Productivity

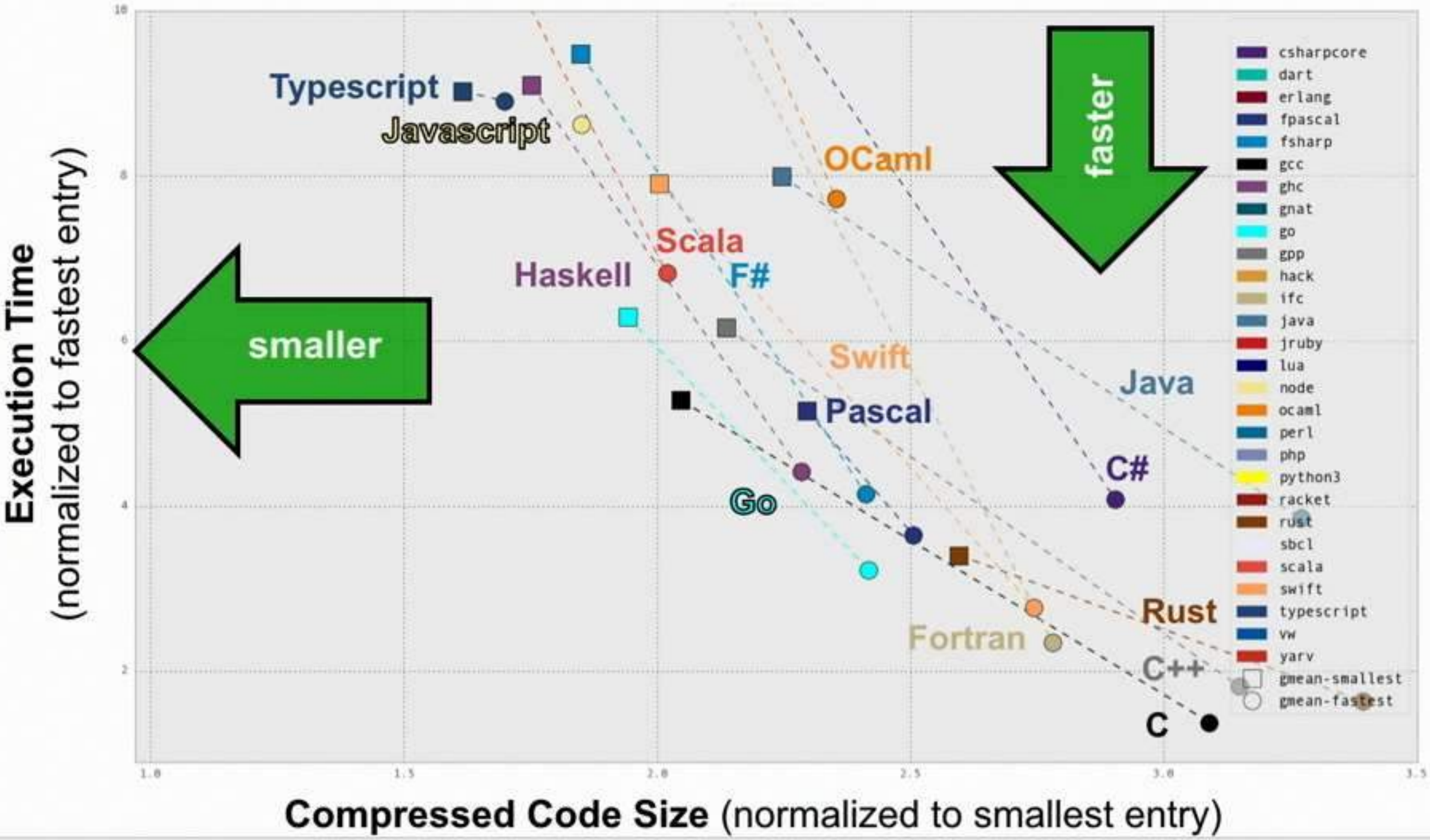
## Chapel aims to be as…

- …**programmable** as Python
- …**fast** as Fortran
- …**scalable** as MPI
- …**portable** as C
- …**flexible** as C++
- …**fun** as [your favorite programming language]

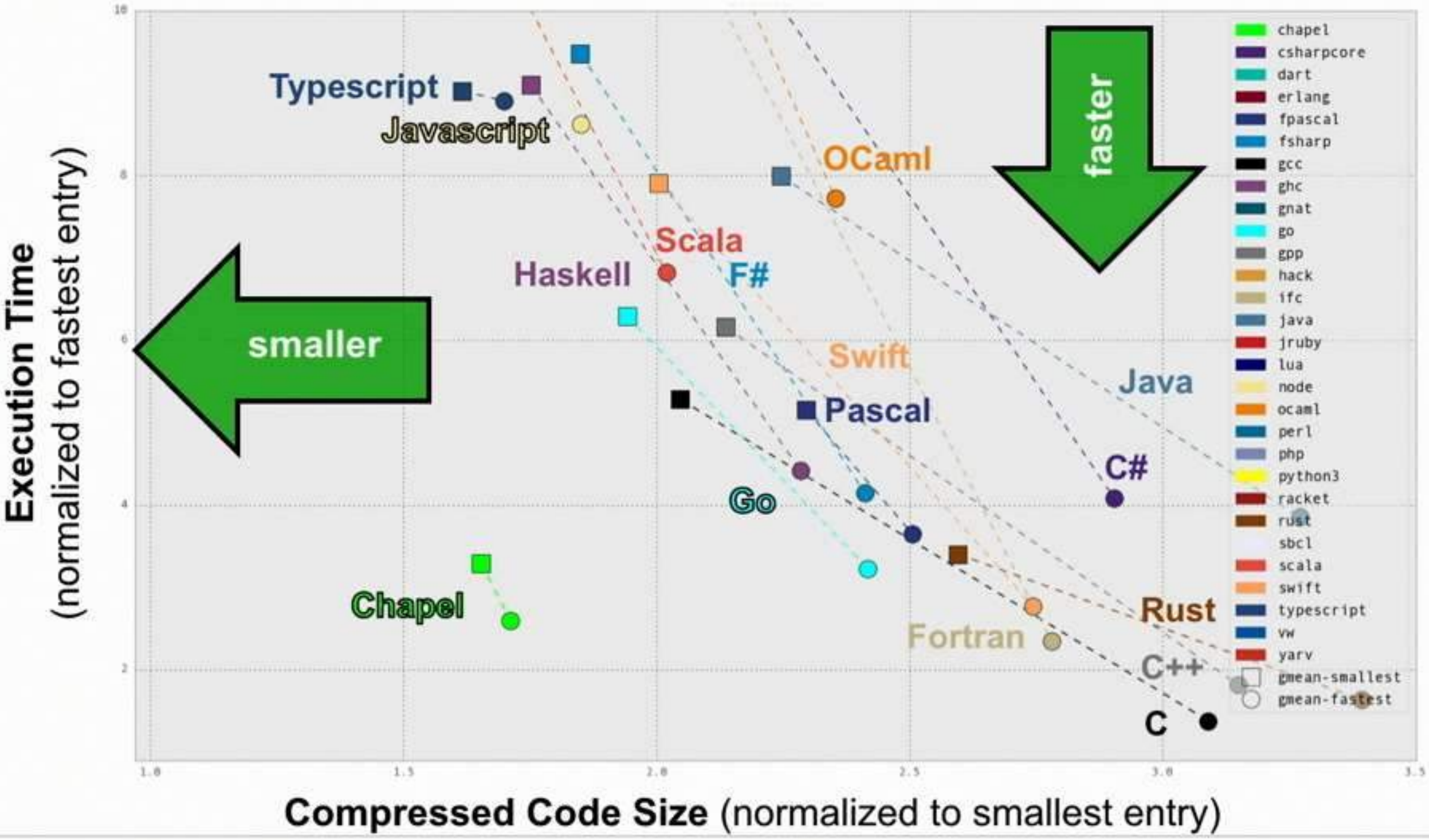
# CLBG Cross-Language Summary
(Oct 2017 standings)



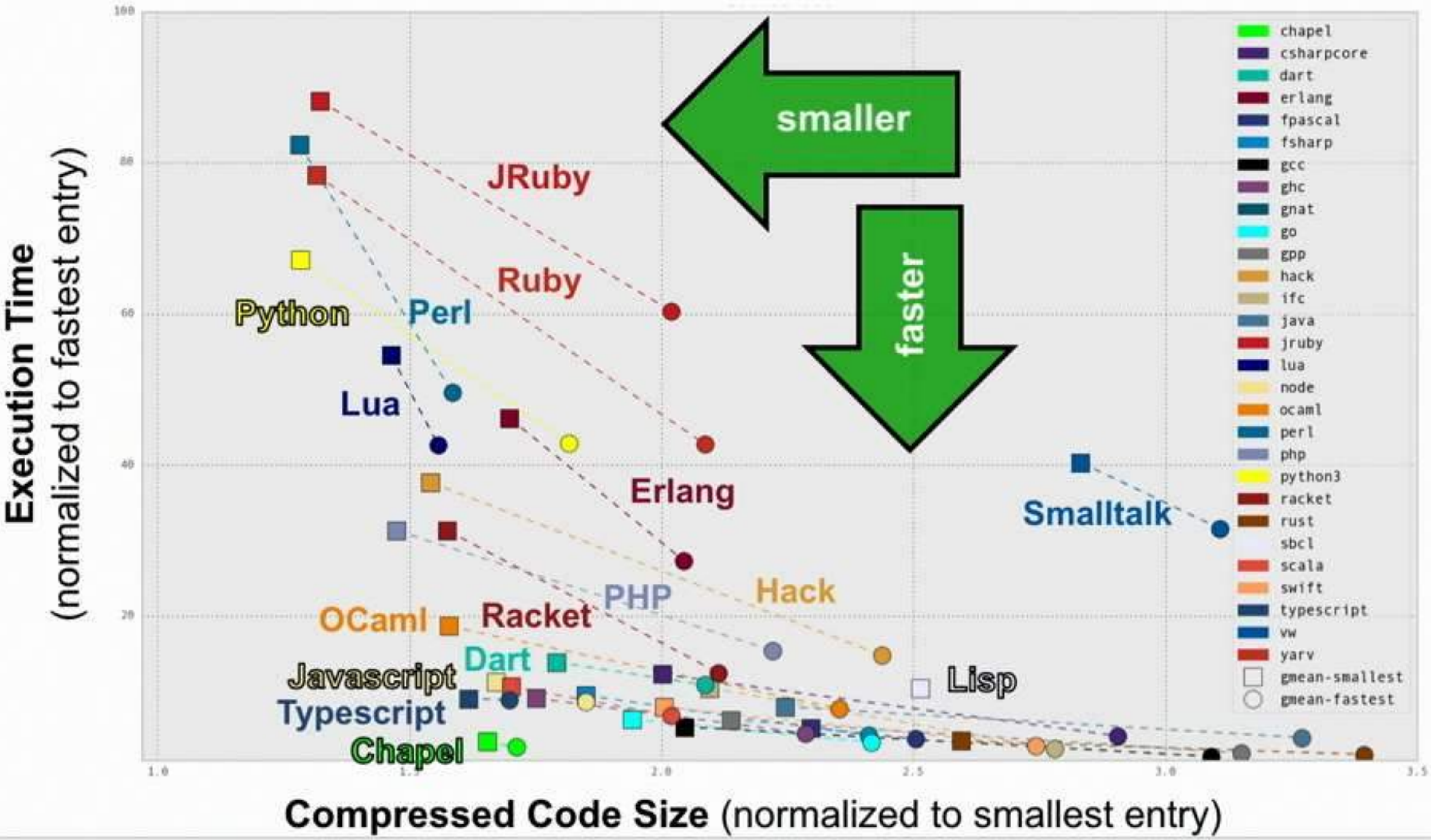
# CLBG Cross-Language Summary
(Oct 2017 standings, zoomed in)



# CLBG Cross-Language Summary
## (Oct 2017 standings, zoomed in)



# CLBG Cross-Language Summary
## (Oct 2017 standings)



# CLBG: Qualitative Code Comparisons

## Can also browse program source code *(but this requires actual thought!)*:

```
proc main() {
  printColorEquations();

  const group1 = [i in 1..popSize1] new Chameneos(i, ((i-1)%3):Color);
  const group2 = [i in 1..popSize2] new Chameneos(i, colors10[i]);

  cobegin {
    holdMeetings(group1, n);
    holdMeetings(group2, n);
  }

  print(group1);
  print(group2);

  for c in group1 do delete c;
  for c in group2 do delete c;
}


//
// Print the results of getNewColor() for all color pairs.
//
proc printColorEquations() {
  for c1 in Color do
    for c2 in Color do
      writeln(c1, " + ", c2, " -> ", getNewColor(c1, c2));
  writeln();
}


//
// Hold meetings among the population by creating a shared meeting
// place, and then creating per-chameneos tasks to have meetings.
//
proc holdMeetings(population, numMeetings) {
  const place = new MeetingPlace(numMeetings);

  coforall c in population do          // create a task per chameneos
    c.haveMeetings(place, population);

  delete place;
}
```

***excerpt from 1210 gz Chapel entry***

```
void get_affinity(int* is_smp, cpu_set_t* affinity1, cpu_set_t* affinity2)
{
    cpu_set_t               active_cpus;
    FILE*                   f;
    char                    buf [2048];
    char const*             pos;
    int                     cpu_idx;
    int                     physical_id;
    int                     core_id;
    int                     cpu_cores;
    int                     apic_id;
    size_t                  cpu_count;
    size_t                  i;

    char const*             processor_str       = "processor";
    size_t                  processor_str_len   = strlen(processor_str);
    char const*             physical_id_str     = "physical id";
    size_t                  physical_id_str_len = strlen(physical_id_str);
    char const*             core_id_str         = "core id";
    size_t                  core_id_str_len     = strlen(core_id_str);
    char const*             cpu_cores_str       = "cpu cores";
    size_t                  cpu_cores_str_len   = strlen(cpu_cores_str);

    CPU_ZERO(&active_cpus);
    sched_getaffinity(0, sizeof(active_cpus), &active_cpus);
    cpu_count = 0;
    for (i = 0; i != CPU_SETSIZE; i += 1)
    {
        if (CPU_ISSET(i, &active_cpus))
        {
            cpu_count += 1;
        }
    }

    if (cpu_count == 1)
    {
        is_smp[0] = 0;
        return;
    }

    is_smp[0] = 1;
    CPU_ZERO(affinity1);
```

***excerpt from 2863 gz C gcc entry***

# CLBG: Qualitative Code Comparisons

## Can also browse program source code *(but this requires actual thought!)*:

```
proc main() {
  printColorEquations();

  const group1 = [i in 1..popSize1] new Chameneos(i, (
  const group2 = [i in 1..popSize2] new Chameneos(i, c

  cobegin {
    holdMeetings(group1, n);
    holdMeetings(group2, n);
  }

  print(group1);
  print(group2);

  for c in group1 do delete c;
  for c in group2 do delete c;
}

//
// Print the results of getNewColor() for all color pa
//
proc printColorEquations() {
  for c1 in Color do
    for c2 in Color do
      writeln(c1, " + ", c2, " -> ", getNewColor(c1, c
  writeln();
}

//
// Hold meetings among the population by creating a sh
// place, and then creating per-chameneos tasks to hav
//
proc holdMeetings(population, numMeetings) {
  const place = new MeetingPlace(numMeetings);

  coforall c in population do          // create a ta
    c.haveMeetings(place, population);

  delete place;
}
```

```
cobegin {
  holdMeetings(group1, n);
  holdMeetings(group2, n);
}
```

```
proc holdMeetings(population, numMeetings) {
  const place = new MeetingPlace(numMeetings);

  coforall c in population do          // creat
    c.haveMeetings(place, population);

  delete place;
}
```

***excerpt from 1210 gz Chapel entry***

```
void get_affinity(int* is_smp, cpu_set_t* affinity1, cpu_set_t* affinity2)
                          active_cpus;
                          f;
                          buf [2048];
                          pos;
                          cpu_idx;
                          physical_id;
                          core_id;
                          cpu_cores;
                          apic_id;
                          cpu_count;
                          i;

                          processor_str       = "processor";
  size_t                  processor_str_len   = strlen(processor_str);
  char const*             physical_id_str     = "physical id";
  size_t                  physical_id_str_len = strlen(physical_id_str);
  char const*             core_id_str         = "core id";
                                              n(core_id_str);
                                              cores";
                                              n(cpu_cores_str);

  is_smp[0] = 1;
  CPU_ZERO(affinity1);
```

***excerpt from 2863 gz C gcc entry***

# CLBG: Qualitative Code Comparisons

Can also browse program source code *(but this requires actual thought!)*:

```
proc main() {
```

```
char const*    core_id_str          = "core id"
size_t         core_id_str_len      = strlen(co
char const*    cpu_cores_str        = "cpu core
size_t         cpu_cores_str_len    = strlen(cp

CPU_ZERO(&active_cpus);
sched_getaffinity(0, sizeof(active_cpus), &active_cpus);
cpu_count = 0;
for (i = 0; i != CPU_SETSIZE; i += 1)
{
    if (CPU_ISSET(i, &active_cpus))
    {
        cpu_count += 1;
    }
}

if (cpu_count == 1)
{
    is_smp[0] = 0;
    return;
}
```

*excerpt from 1210 gz Chapel entry*

```
void get_affinity(int* is_smp, cpu_set_t* affinity1, cpu_set_t* affinity2)
{
    cpu_set_t        active_cpus;
    FILE*            f;
    char             buf [2048];
    char const*      pos;
    int              cpu_idx;
    int              physical_id;
    int              core_id;
    int              cpu_cores;
    int              apic_id;
    size_t           cpu_count;
    size_t           i;

    char const*      processor_str         = "processor";
    size_t           processor_str_len     = strlen(processor_str);
    char const*      physical_id_str       = "physical id";
    size_t           physical_id_str_len   = strlen(physical_id_str);
    char const*      core_id_str           = "core id";
    size_t           core_id_str_len       = strlen(core_id_str);
    char const*      cpu_cores_str         = "cpu cores";
    size_t           cpu_cores_str_len     = strlen(cpu_cores_str);

    CPU_ZERO(&active_cpus);
    sched_getaffinity(0, sizeof(active_cpus), &active_cpus);
    cpu_count = 0;
    for (i = 0; i != CPU_SETSIZE; i += 1)
    {
        if (CPU_ISSET(i, &active_cpus))
        {
            cpu_count += 1;
        }
    }

    if (cpu_count == 1)
    {
        is_smp[0] = 0;
        return;
    }

    is_smp[0] = 1;
    CPU_ZERO(affinity1);
```

*excerpt from 2863 gz C gcc entry*

# Excerpt from PNW PLSE Review

*"Chapel has been around for quite a while, and it still seems like a niche language…"*

# Chapel: “A Niche Language”?

Chapel is arguably niche in that it…

…was originally designed for HPC

# Chapel: “A Niche Language”?

Chapel is arguably niche in that it…

…was originally designed for HPC

Yet, Chapel’s chief concerns aren’t HPC-specific:

- performance

# Chapel: “A Niche Language”?

Chapel is arguably niche in that it…

…was originally designed for HPC

Yet, Chapel’s chief concerns aren’t HPC-specific:

- performance
- programmability (cf. Python)

# Chapel: "A Niche Language"?

Chapel is arguably niche in that it…

…was originally designed for HPC

Yet, Chapel's chief concerns aren't HPC-specific:

- performance
- programmability (cf. Python)
- parallelism (cf. multicore)

# Chapel: "A Niche Language"?

Chapel is arguably niche in that it…

…was originally designed for HPC

Yet, Chapel's chief concerns aren't HPC-specific:

- performance
- programmability (cf. Python)
- parallelism (cf. multicore)
- distributed memory (cf. cloud computing)

# Chapel: "A Niche Language"?

Chapel is arguably niche in that it…

- …was originally designed for HPC
- …has only a modest-sized community (so far)

# Chapel: "A Niche Language"?

Chapel is arguably niche in that it…

- …was originally designed for HPC
- …has only a modest-sized community (so far)

Yet, we've historically discouraged its use in production…

# Chapel: A Quiche Language!

The outsider's impression:

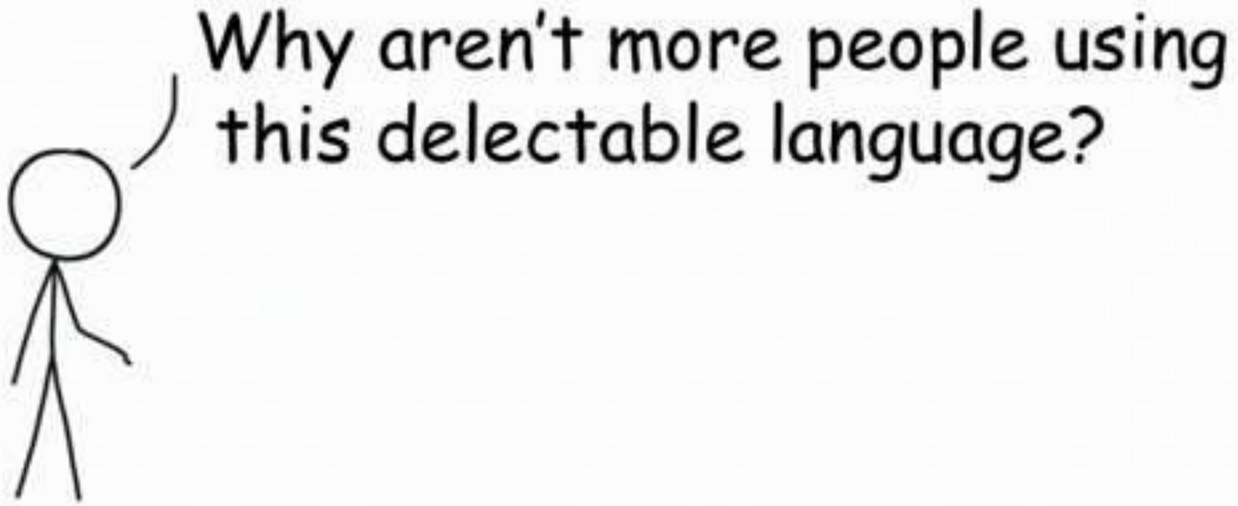



Image sources: https://www.themountainkitchen.com, https://xkcd.com/

# Chapel: A Quiche Language!

The outsider's impression:

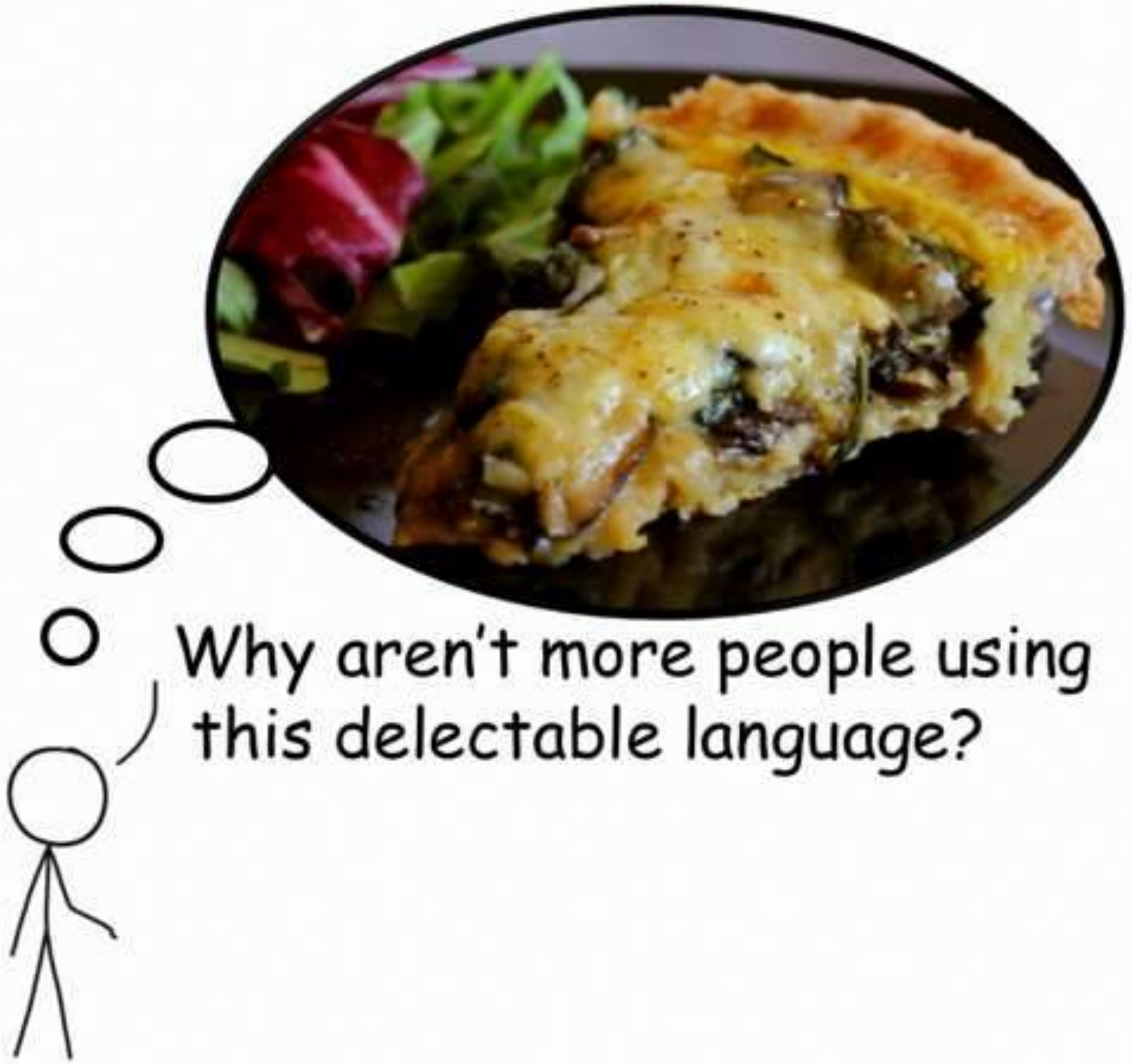
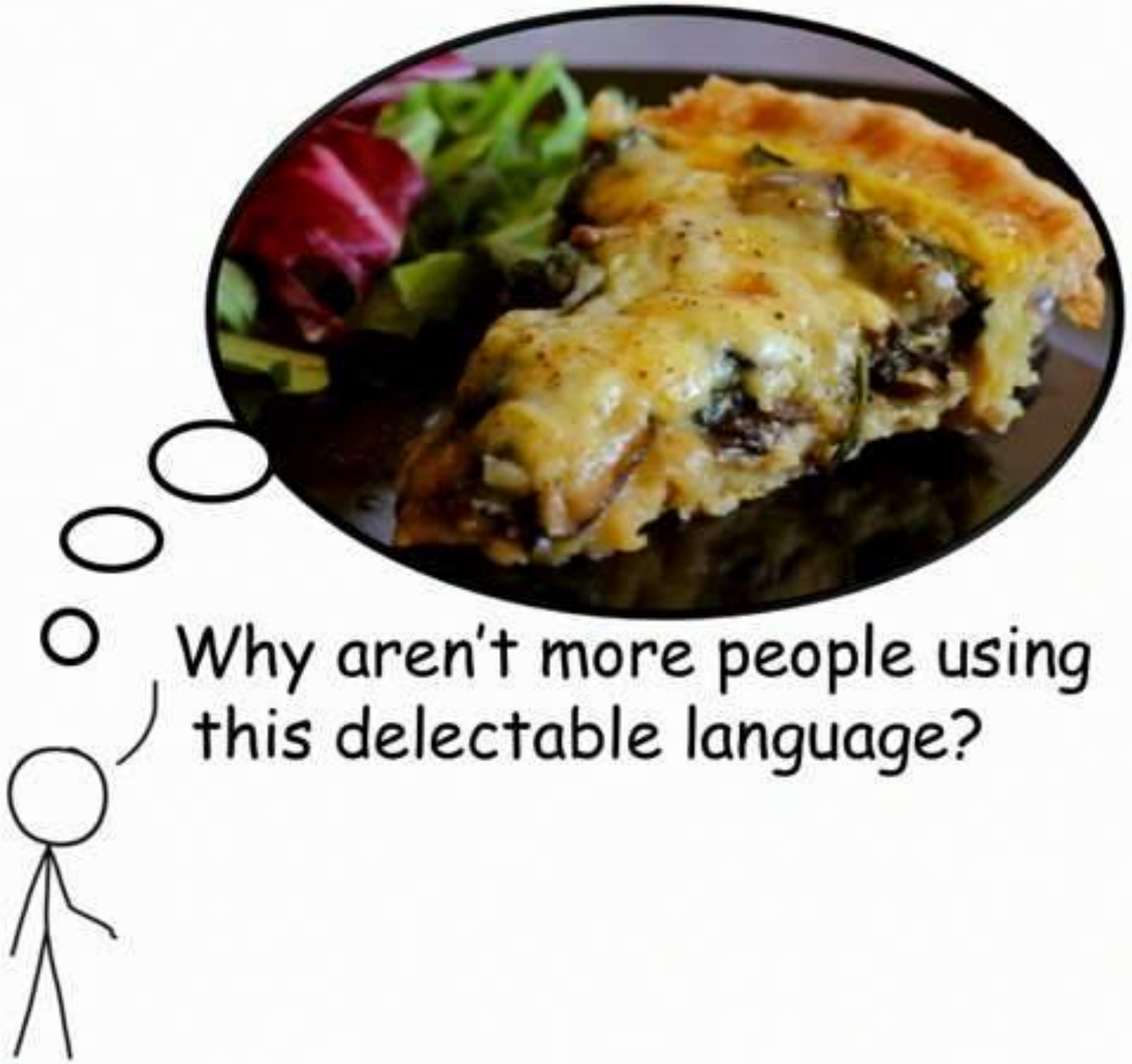





Image sources: https://www.themountainkitchen.com, https://xkcd.com/

# Chapel: A Quiche Language!

The outsider's impression:



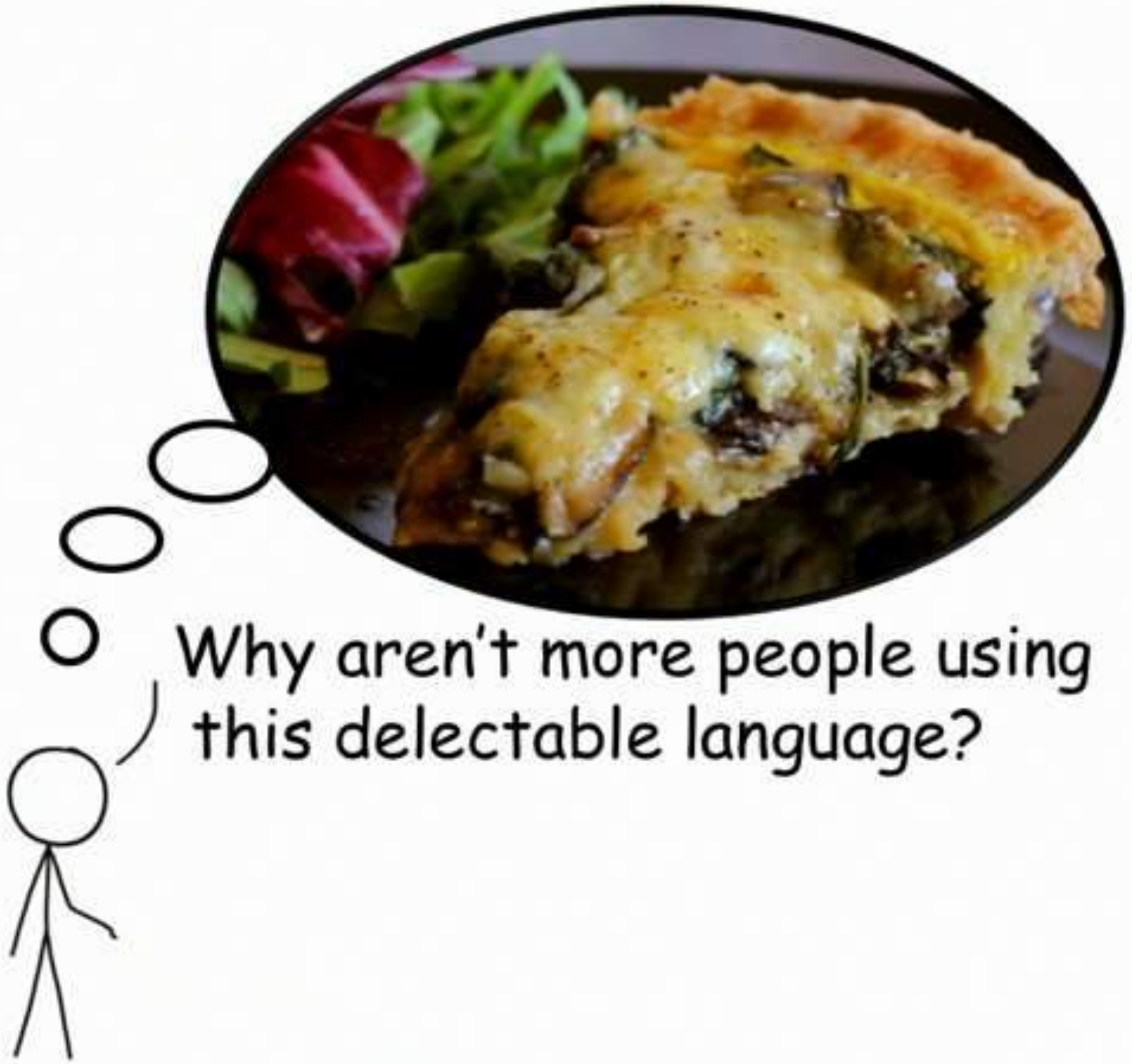

The reality, for most of Chapel's history:



Image sources: https://www.themountainkitchen.com, https://xkcd.com/

# Chapel: A Quiche Language!

The outsider's impression:




Though recently, it's more like:

Image sources: https://www.themountainkitchen.com, https://xkcd.com/

# Chapel: "Been Around for Quite Awhile"

## **Chapel's Infancy:** DARPA HPCS (2003–2012)

- Research focus: ~6-7 FTEs
  - distinguish locality from parallelism
  - seamlessly mix data- and task-parallelism
  - support user-defined distributed arrays, parallel iterators

# Chapel: "Been Around for Quite Awhile"

## **Chapel's Infancy:** DARPA HPCS (2003–2012)

- Research focus: ~6-7 FTEs
  - distinguish locality from parallelism
  - seamlessly mix data- and task-parallelism
  - support user-defined distributed arrays, parallel iterators

## **Chapel's Adolescence:** "the five-year push" (2013–2018)

- Development focus: ~13-14 FTEs
  - **performance and scalability**
  - **ecosystem:** documentation, libraries, tools, …
  - **base language fixes:** OOP features, error-handling, strings, …

# Chapel: "Been Around for Quite Awhile"

## Chapel's Infancy: DARPA HPCS (2003–2012)

- Research focus: ~6-7 FTEs
  - distinguish locality from parallelism
  - seamlessly mix data- and task-parallelism
  - support user-defined distributed arrays, parallel iterators

## Chapel's Adolescence: "the five-year push" (2013–2018)

Then Now

- Development focus: ~13-14 FTEs
  - **performance and scalability**
  - **ecosystem:** documentation, libraries, tools, …
  - **base language fixes:** OOP features, error-handling, strings, …

# Chapel Ecosystem: Then vs. Now

# Documentation: Then

## After HPCS:

- a PDF language specification


Chapel Language Specification
Version 0.93

Cray Inc
901 Fifth Avenue, Suite 1000
Seattle, WA 98164

April 18, 2013

# Documentation: Then

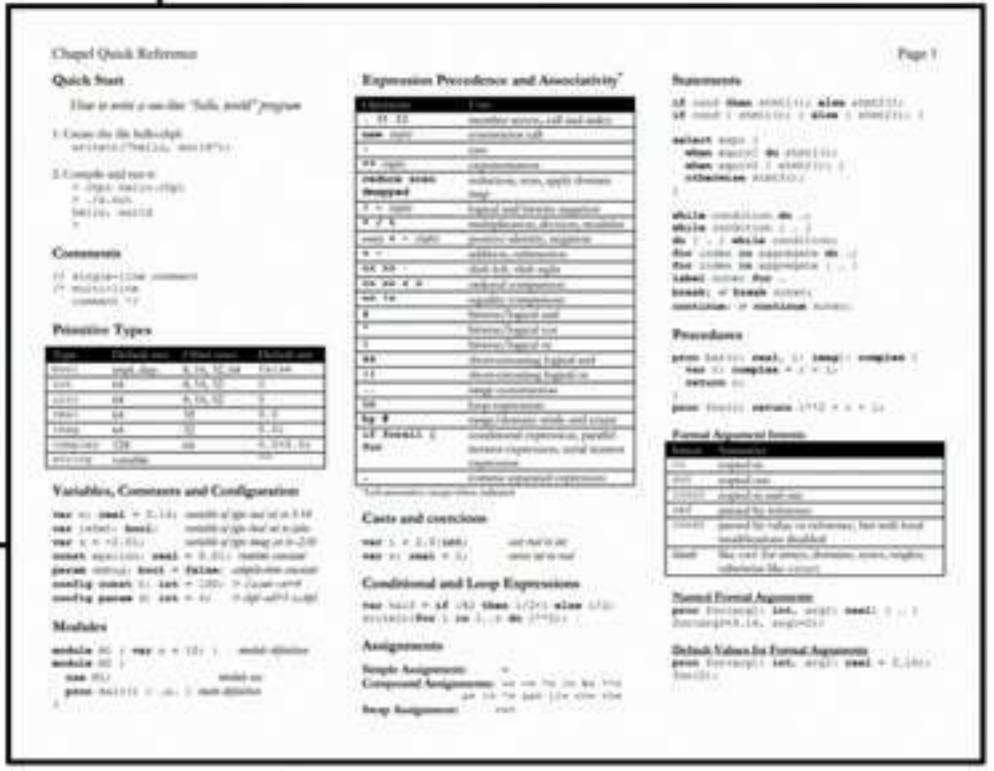
## After HPCS:

- a PDF language specification
- a Quick Reference sheet


Chapel Language Specification
Version 0.93

Cray Inc
901 Fifth Avenue, Suite 1000
Seattle, WA 98164

April 18, 2013



# Documentation: Then

## After HPCS:

- a PDF language specification
- a Quick Reference sheet
- a number of READMEs

Chapel Language Specification
Version 0.93

Cray Inc
901 Fifth Avenue, Suite 1000
Seattle, WA 98164

April 18, 2013

Chapel Quick Reference — Page 1

Quick Start

Comments

Primitive Types

Variables, Constants and Configuration

Modules

Expression Precedence and Associativity

Casts and coercions

Conditional and Loop Expressions

Assignments

Statements

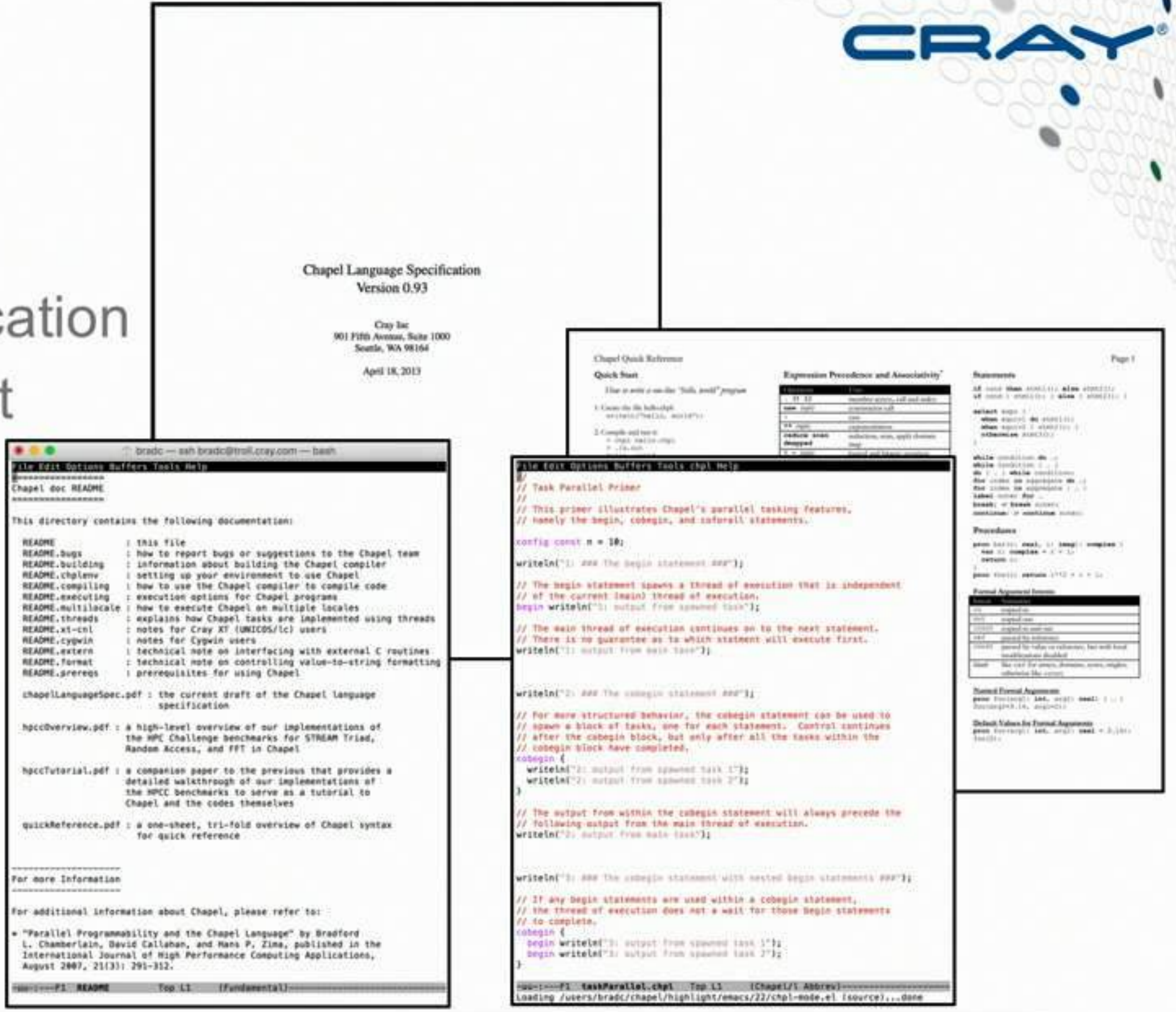
Procedures

```
bradc — ssh bradc@troll.cray.com — bash
File Edit Options Buffers Tools Help
=================
Chapel doc README
=================

This directory contains the following documentation:

  README              : this file
  README.bugs         : how to report bugs or suggestions to the Chapel team
  README.building     : information about building the Chapel compiler
  README.chplenv      : setting up your environment to use Chapel
  README.compiling    : how to use the Chapel compiler to compile code
  README.executing    : execution options for Chapel programs
  README.multilocale  : how to execute Chapel on multiple locales
  README.threads      : explains how Chapel tasks are implemented using threads
  README.xt-cnl       : notes for Cray XT (UNICOS/lc) users
  README.cygwin       : notes for Cygwin users
  README.extern       : technical note on interfacing with external C routines
  README.format       : technical note on controlling value-to-string formatting
  README.prereqs      : prerequisites for using Chapel

  chapelLanguageSpec.pdf : the current draft of the Chapel language
                           specification

  hpccOverview.pdf : a high-level overview of our implementations of
                     the HPC Challenge benchmarks for STREAM Triad,
                     Random Access, and FFT in Chapel

  hpccTutorial.pdf : a companion paper to the previous that provides a
                     detailed walkthrough of our implementations of
                     the HPCC benchmarks to serve as a tutorial to
                     Chapel and the codes themselves

  quickReference.pdf : a one-sheet, tri-fold overview of Chapel syntax
                       for quick reference


--------------------
For more Information
--------------------

For additional information about Chapel, please refer to:

* "Parallel Programmability and the Chapel Language" by Bradford
  L. Chamberlain, David Callahan, and Hans P. Zima, published in the
  International Journal of High Performance Computing Applications,
  August 2007, 21(3): 291-312.

-uu-:----F1  README        Top L1     (Fundamental)--------------------
```

# Documentation: Then

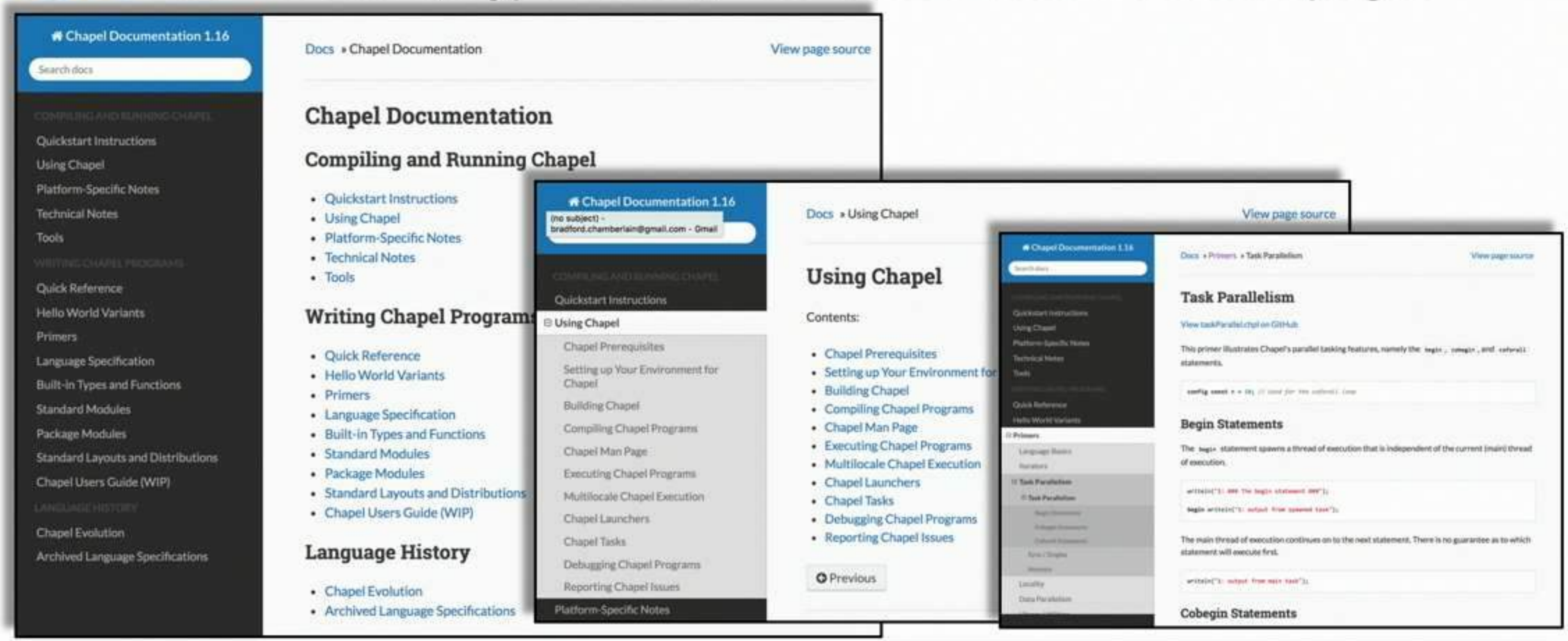
## After HPCS:

- a PDF language specification
- a Quick Reference sheet
- a number of READMEs
- ~22 primer examples



# Documentation: Now

**Now:** 200+ modern, hyperlinked, web-based documentation pages



# Libraries: Then

## **After HPCS:** ~25 library modules

- documented via source comments, if at all:

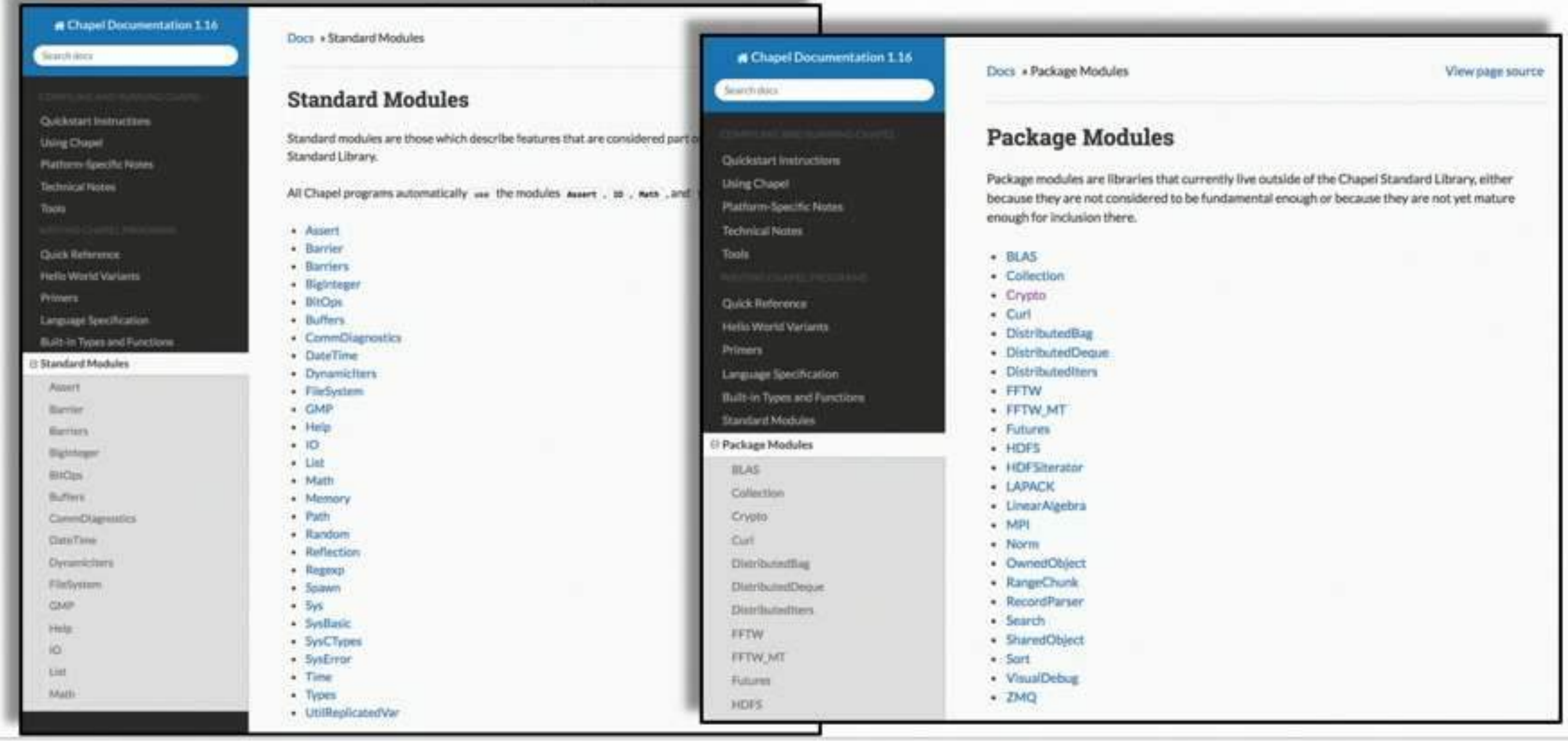
```
bradc — ssh bradc@troll.cray.com — bash
File Edit Options Buffers Tools chpl Help
// Copyright (c) 2004-2013, Cray Inc.  (See LICENSE file for more details)


//
// Random Module
//
// This standard module contains a random number generator based on
// the one used in the NPB benchmarks.  Tailoring the NPB comments to
// this code, we can say the following:
//
//   This generator returns uniform pseudorandom real values in the
//   range (0, 1) by using the linear congruential generator
//
//     x_{k+1} = a x_k  (mod 2**46)
//
//   where 0 < x_k < 2**46 and 0 < a < 2**46.  This scheme generates
//   2**44 numbers before repeating.  The seed value must be an odd
//   64-bit integer in the range (1, 2^46).  The generated values are
//   normalized to be between 0 and 1, i.e., 2**(-46) * x_k.
//
//   This generator should produce the same results on any computer
//   with at least 48 mantissa bits for real(64) data.
//
// Open Issues
//
// 1. We would like to support general serial and parallel iterators
// on the RandomStream class, but this is not possible with our
// current parallel iterator framework.
//
// 2. The random number generation functionality in this module is
// currently restricted to 64-bit real, 64-bit imag, and 128-bit
// complex values.  This should be extended to other primitive types
// for which this would make sense.  Coercions are insufficient.
//
// 3. Can the multiplier 'arand' be moved into the RandomStream class
// so that it can be changed by a user of this class.
//
// 4. By default, the random stream seed is initialized based on the
// current time in microseconds, allowing for some degree of
// randomness.  The intent of the SeedGenerator enumerated type is to
// provide a menu of options for initializing the random stream seed,
// but only one option is implemented to date.
//
// Note on Private
//
// It is the intent that once Chapel supports the notion of 'private',
// everything prefixed with RandomPrivate_ will be made private to
-uu-:---F1  Random.chpl    Top L1     (Chapel/l Abbrev)----------------------
Mark set
```

```
bradc — ssh bradc@troll.cray.com — bash
File Edit Options Buffers Tools chpl Help
// Copyright (c) 2004-2013, Cray Inc.  (See LICENSE file for more details)


extern type qio_regexp_t;

extern record qio_regexp_options_t {
  var utf8:bool;
  var posix:bool;
  var literal:bool;
  var nocapture:bool;
  // These ones can be set inside the regexp
  var ignorecase:bool; // (?i)
  var multiline:bool; // (?m)
  var dotnl:bool; // (?s)
  var nongreedy:bool; // (?U)
}

extern proc qio_regexp_null():qio_regexp_t;
extern proc qio_regexp_init_default_options(ref options:qio_regexp_options_t);
extern proc qio_regexp_create_compile(str:string, strlen:int(64), ref options:q\
io_regexp_options_t, ref compiled:qio_regexp_t);
extern proc qio_regexp_create_compile_flags(str:string, strlen:int(64), flags:s\
tring, flagslen:int(64), isUtf8:bool, ref compiled:qio_regexp_t);
extern proc qio_regexp_create_compile_flags_2(str:c_ptr, strlen:int(64), flags:\
c_ptr, flagslen:int(64), isUtf8:bool, ref compiled:qio_regexp_t);
extern proc qio_regexp_retain(ref compiled:qio_regexp_t);
extern proc qio_regexp_release(ref compiled:qio_regexp_t);

extern proc qio_regexp_get_options(ref regexp:qio_regexp_t, ref options: qio_re\
gexp_options_t);
extern proc qio_regexp_get_pattern(ref regexp:qio_regexp_t, ref pattern: string\
);
extern proc qio_regexp_get_ncaptures(ref regexp:qio_regexp_t):int(64);
extern proc qio_regexp_ok(ref regexp:qio_regexp_t):bool;
extern proc qio_regexp_error(ref regexp:qio_regexp_t):string;

extern const QIO_REGEXP_ANCHOR_UNANCHORED:c_int;
extern const QIO_REGEXP_ANCHOR_START:c_int;
extern const QIO_REGEXP_ANCHOR_BOTH:c_int;

extern record qio_regexp_string_piece_t {
  var offset:int(64); // counting from 0, -1 means "NULL"
  var len:int(64);
}

extern proc qio_regexp_string_piece_isnull(ref sp:qio_regexp_string_piece_t):bo\
ol;
-uu-:---F1  Regexp.chpl    Top L1     (Chapel/l Abbrev)----------------------
```

# Libraries: Now

## **Now:** ~60 library modules

- web-documented, many user-contributed



# Tools: Then

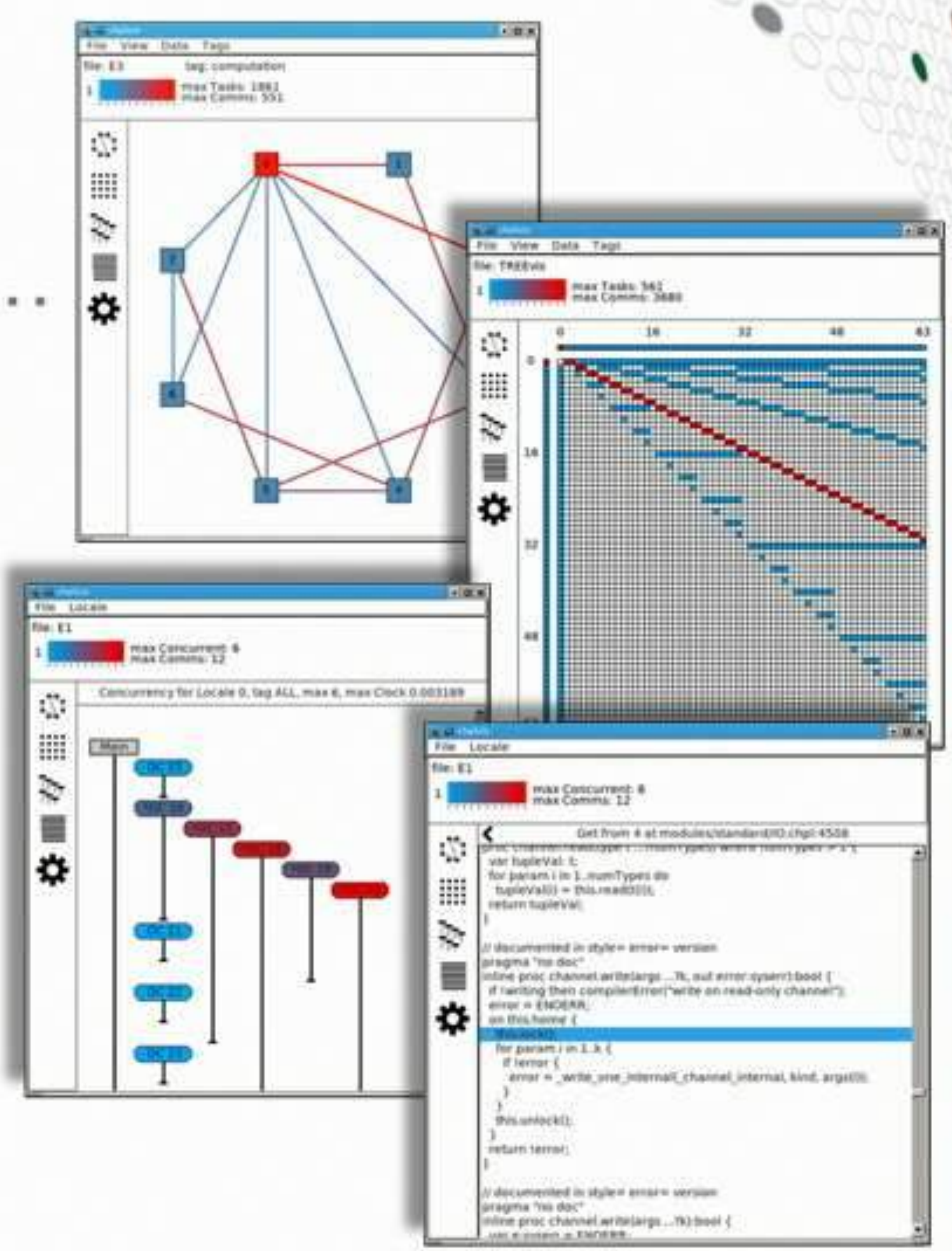
## After HPCS:

- **highlighting modes** for emacs and vim
- **chpldoc:** documentation tool (rough draft)

# Tools: Now

## Now:

- **highlighting modes** for emacs, vim, atom, …
- **chpldoc:** documentation tool
- **mason:** package manager
- **c2chapel:** interoperability aid
- **chpltags:** helps search Chapel code
- **bash tab completion:** command-line help
- **chplvis:** performance visualizer / debugger

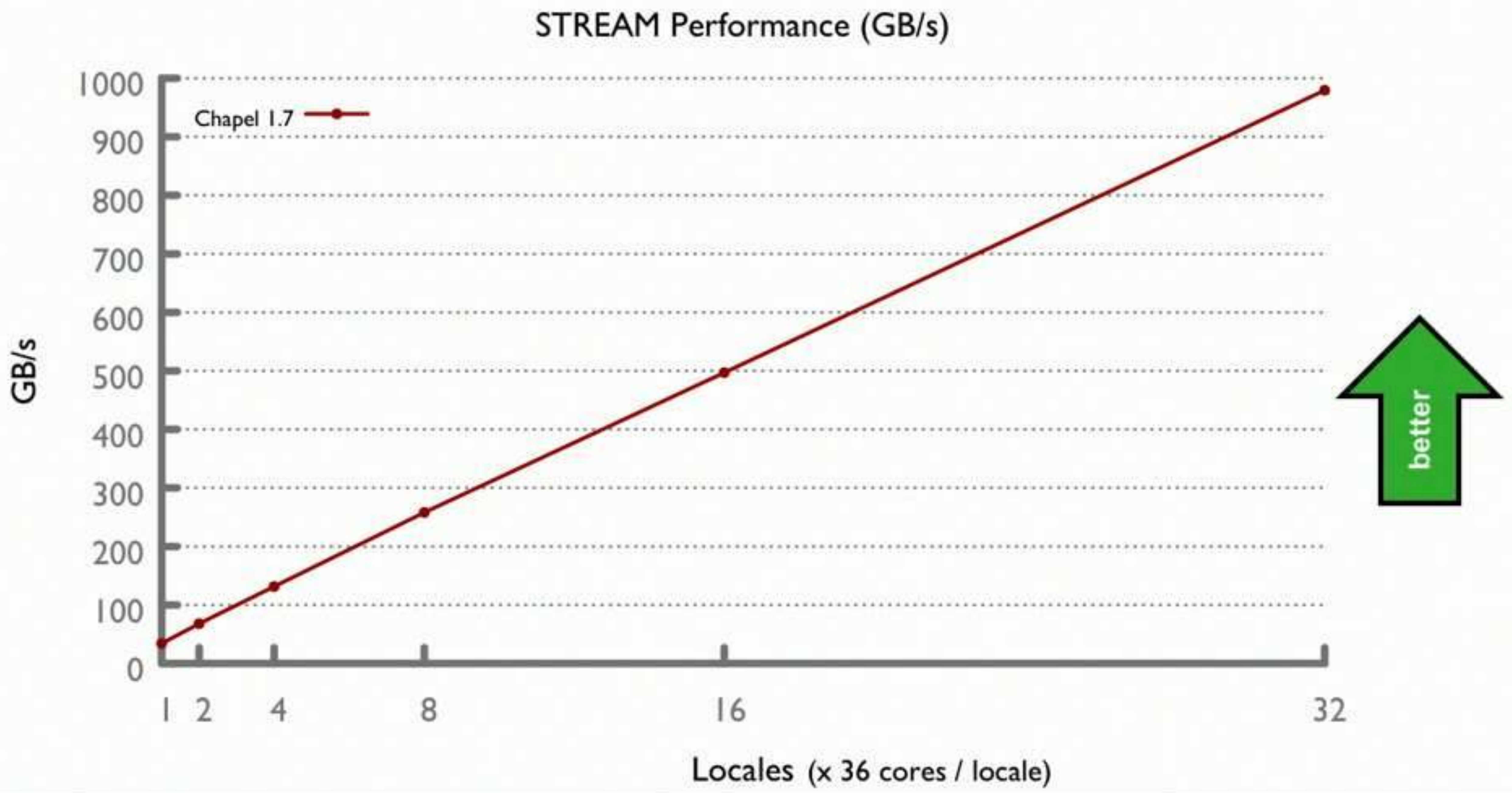


# Chapel Performance: Then vs. Now

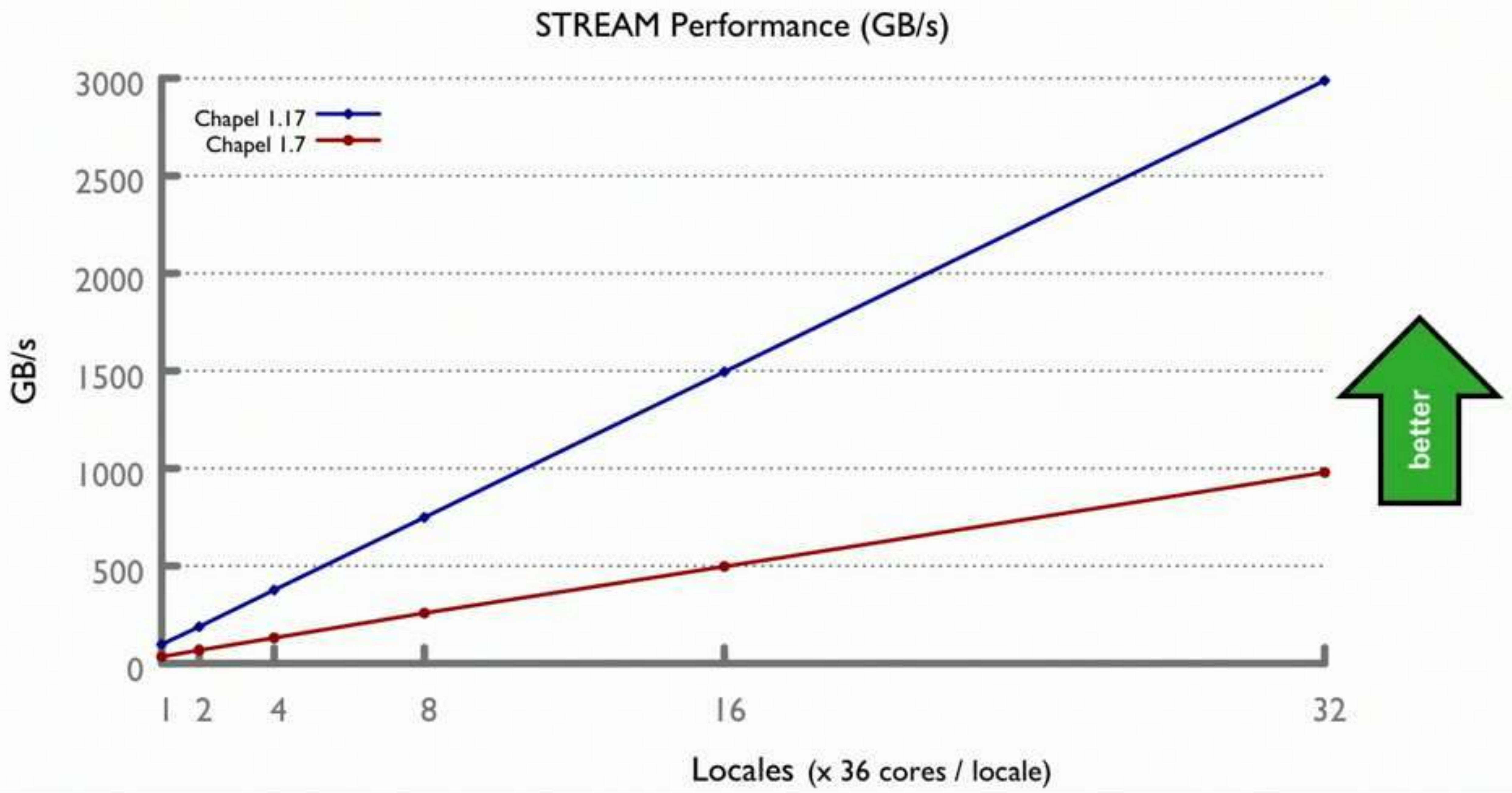
# Performance Focus Areas during 5-year push

- Cleaner, simpler generated code
- NUMA sensitivity within multi-socket nodes
- Best-use of RDMA and NIC memory registration
- Reduced overheads in tasks, memory, communication
- Bulk transfer optimizations
- …and much more…

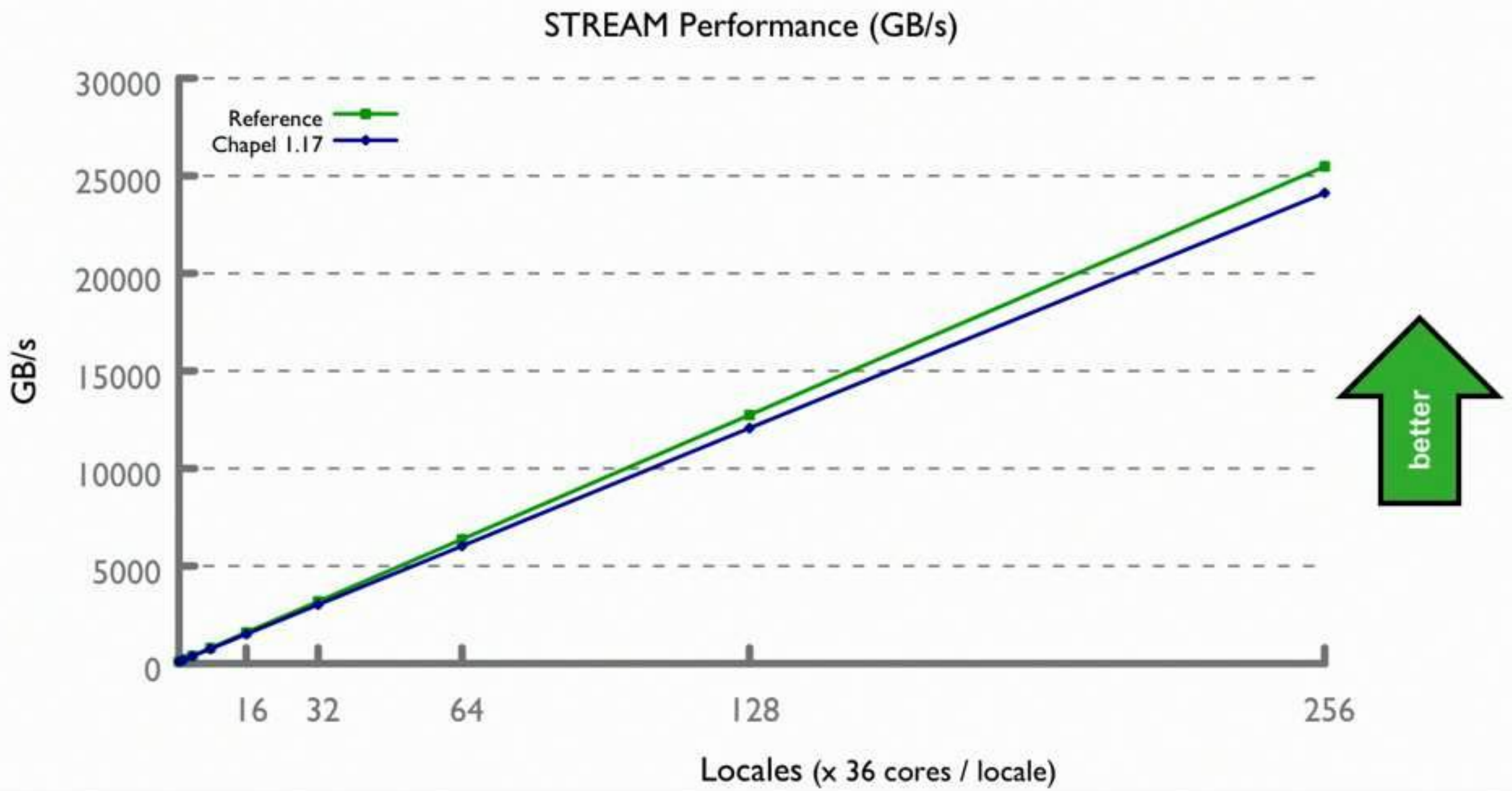
# STREAM Triad Performance: Chapel Then



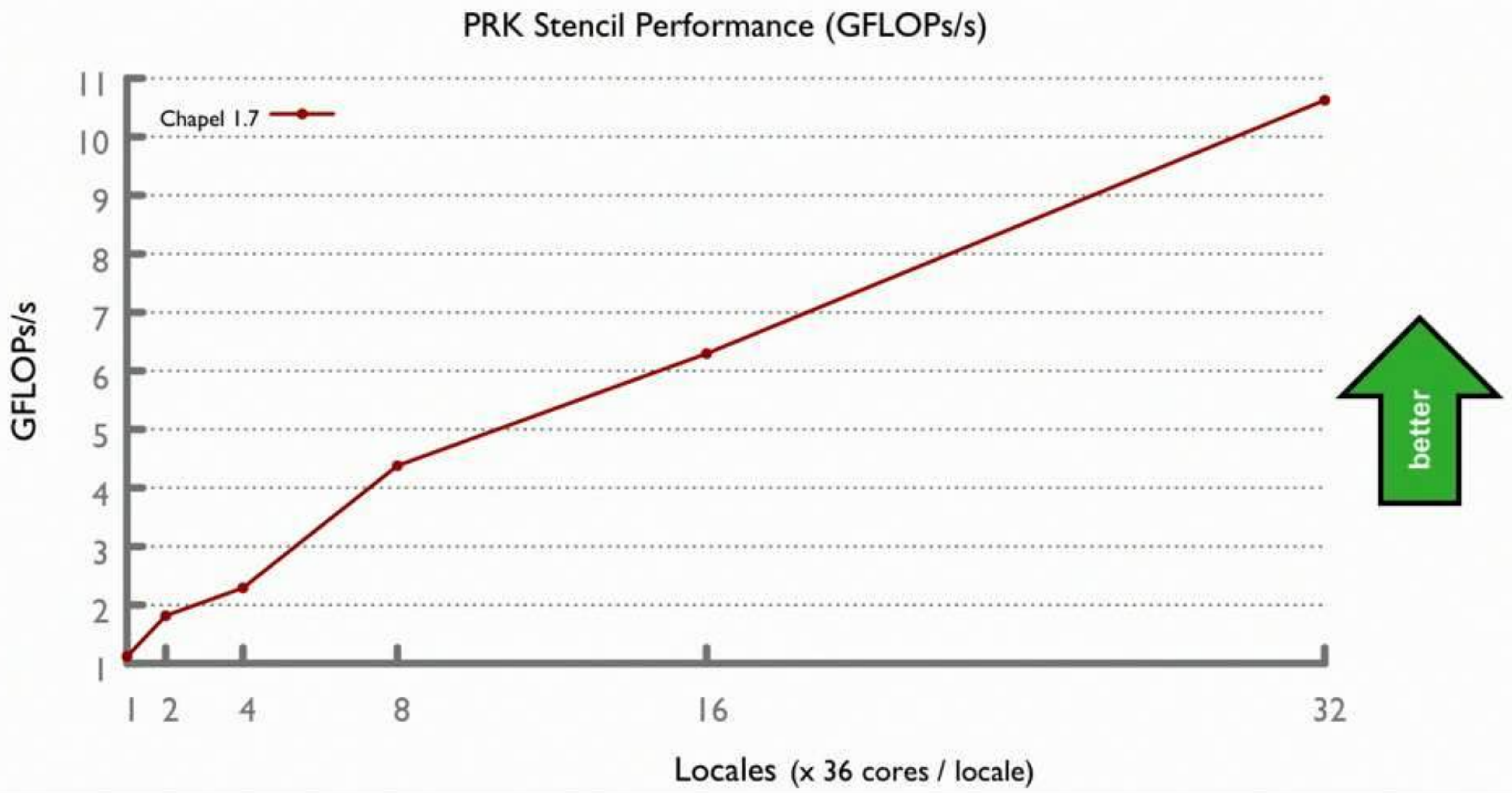
# STREAM Triad Performance: Chapel Then vs. Now



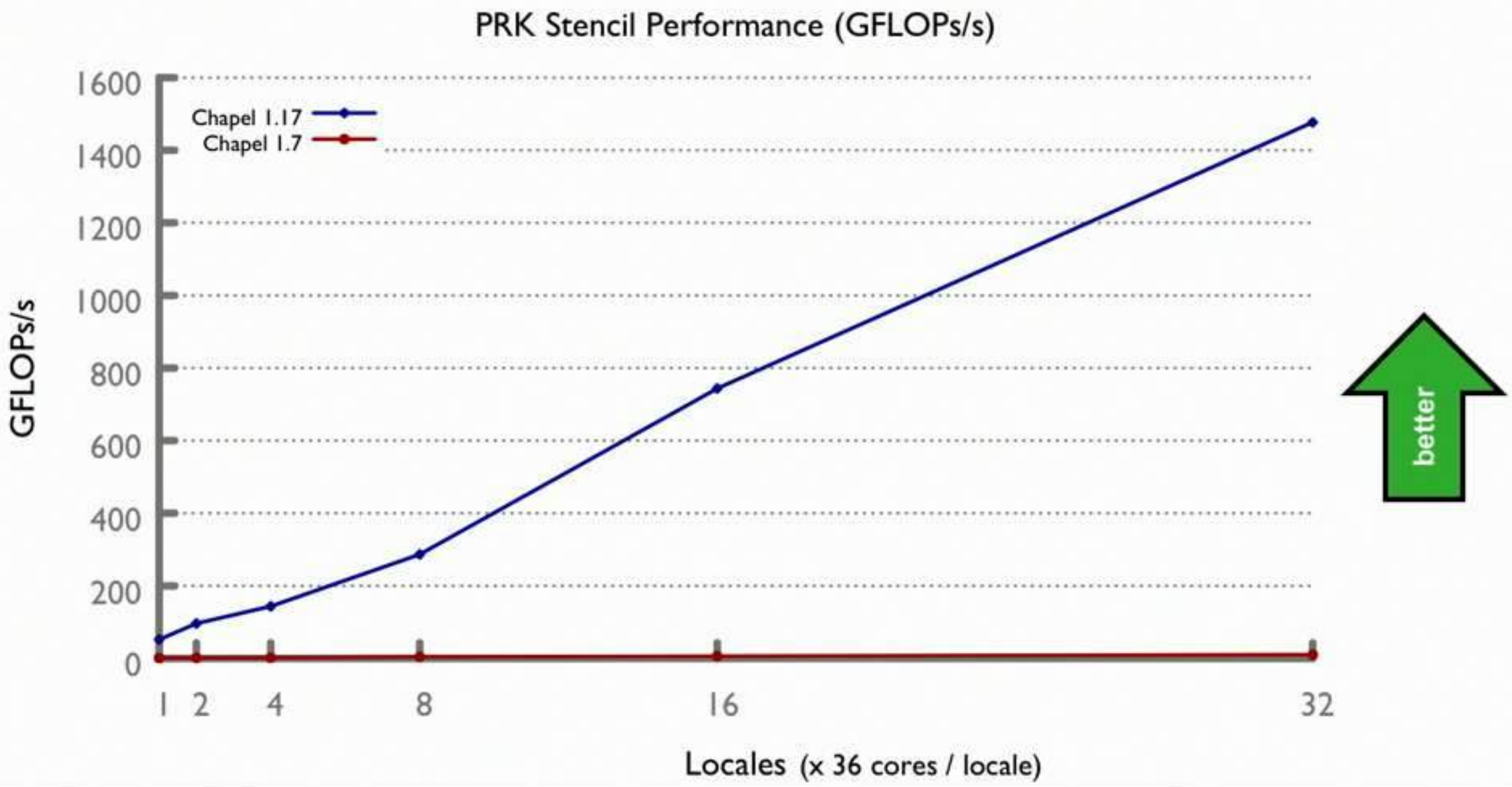
# STREAM Triad Performance: Chapel Now vs. ref



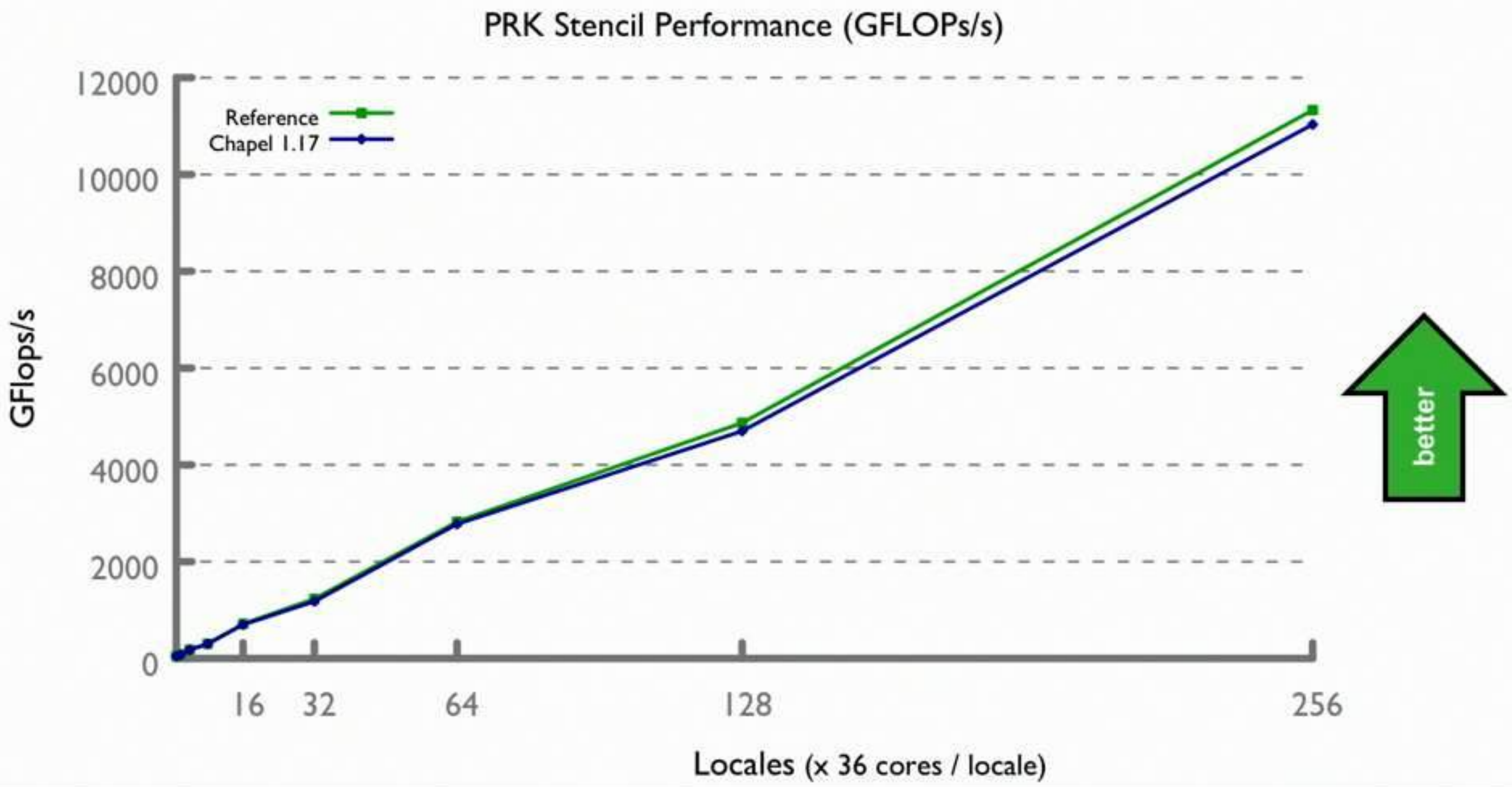
# PRK Stencil Performance: Chapel Then



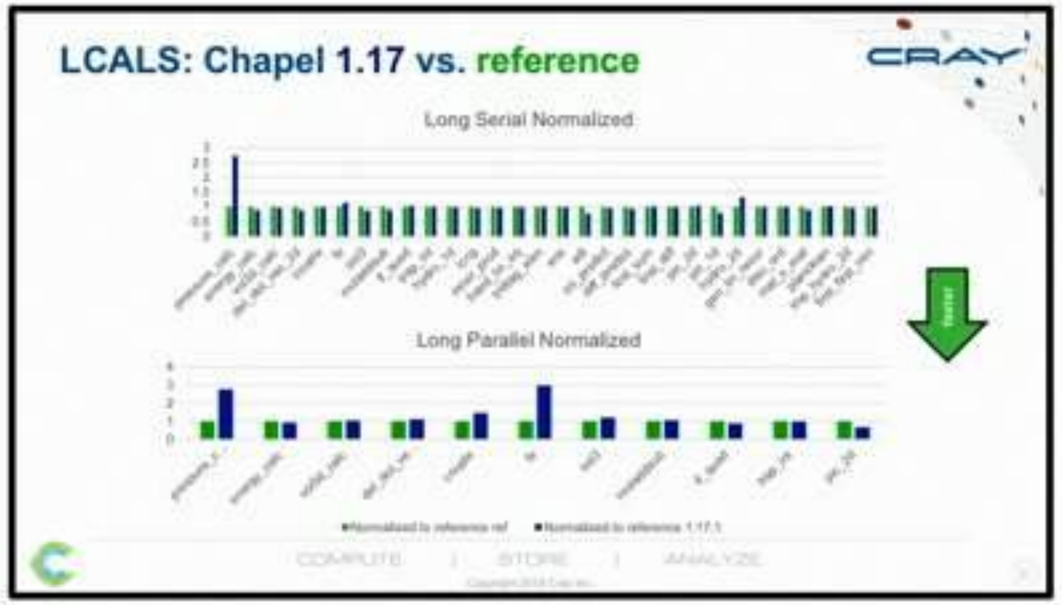
# PRK Stencil Performance: Chapel Then vs. Now



# PRK Stencil Performance: Chapel Now vs. ref

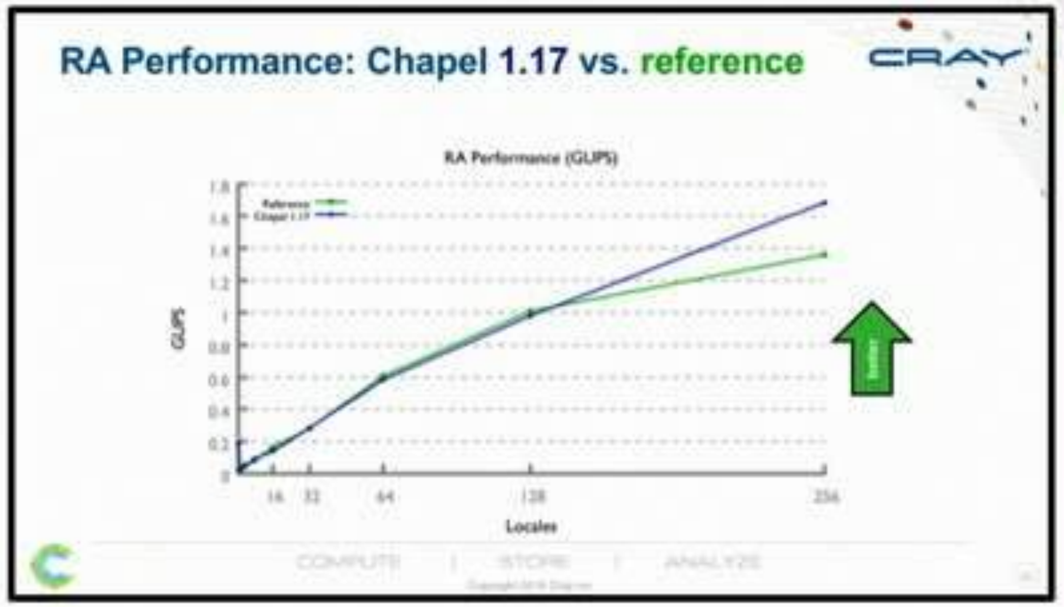


# HPC Patterns: Chapel Now vs. reference




LCALS

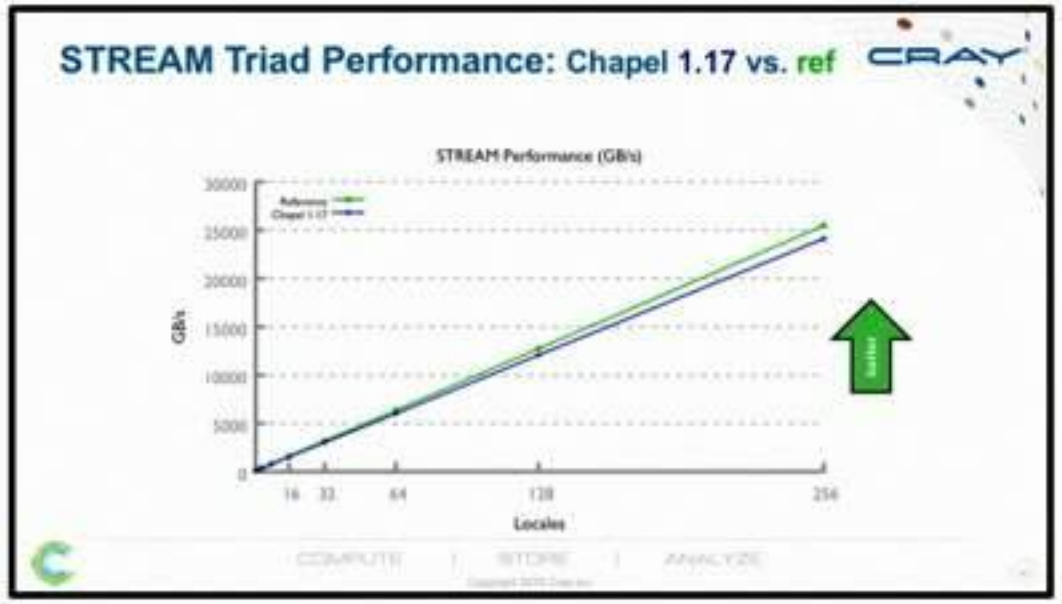
HPCC RA

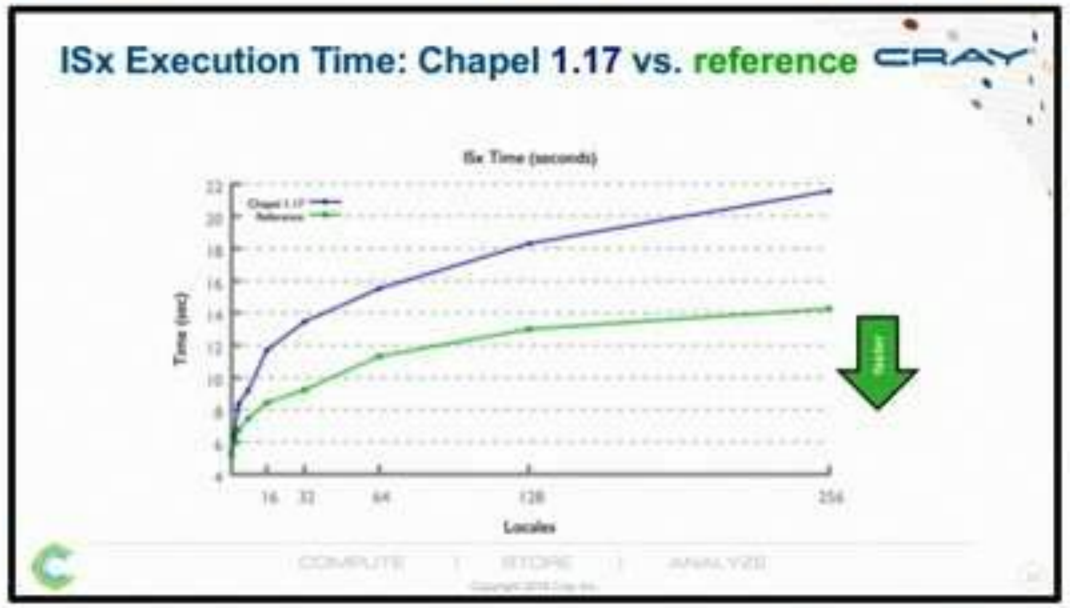


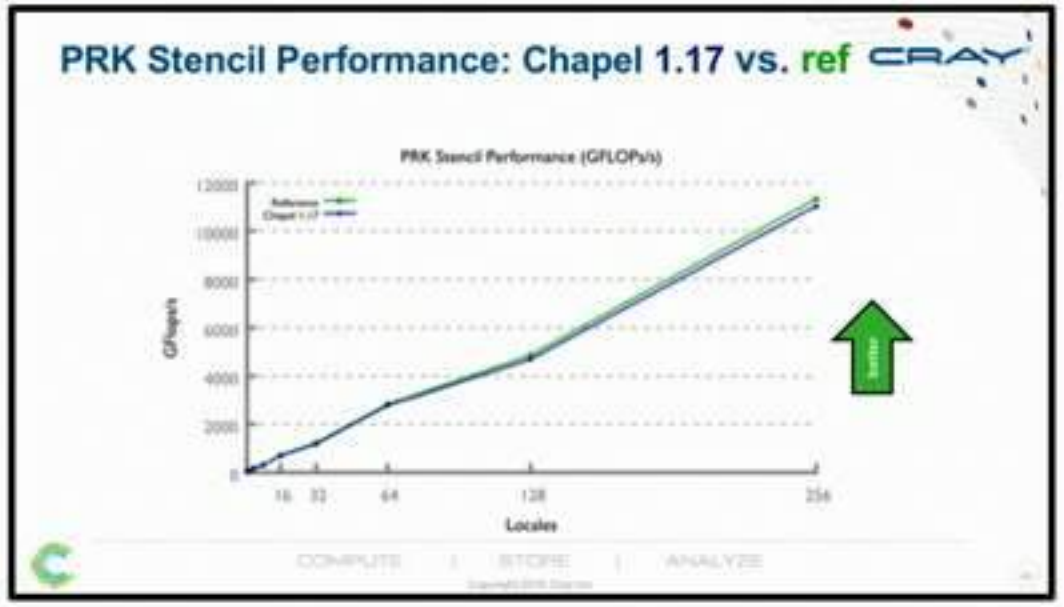

STREAM Triad

ISx

PRK Stencil

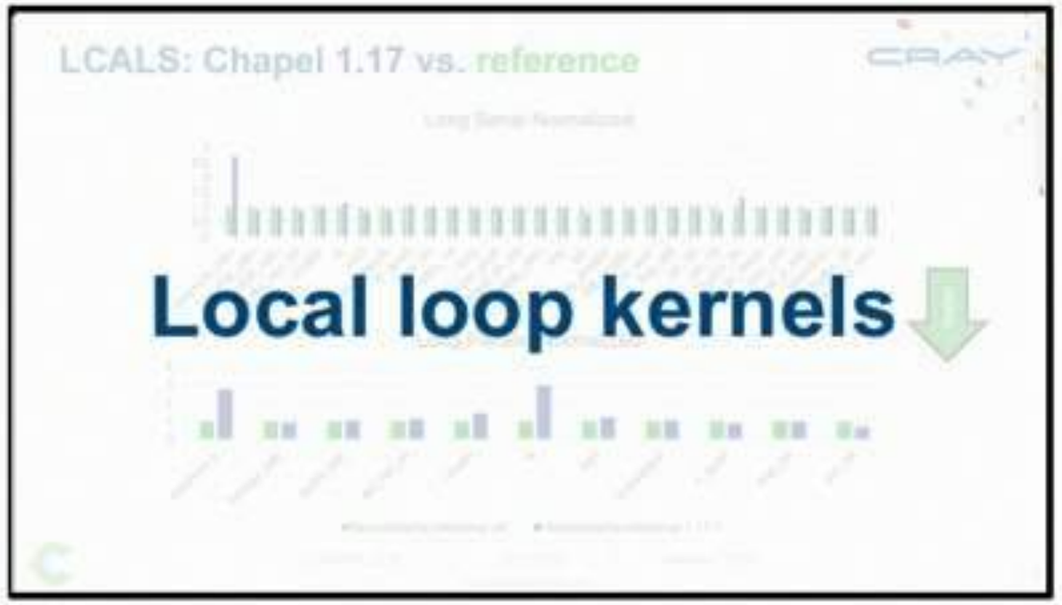



Nightly performance tickers online at:
https://chapel-lang.org/perf-nightly.html

# HPC Patterns: Chapel Now vs. reference




LCALS

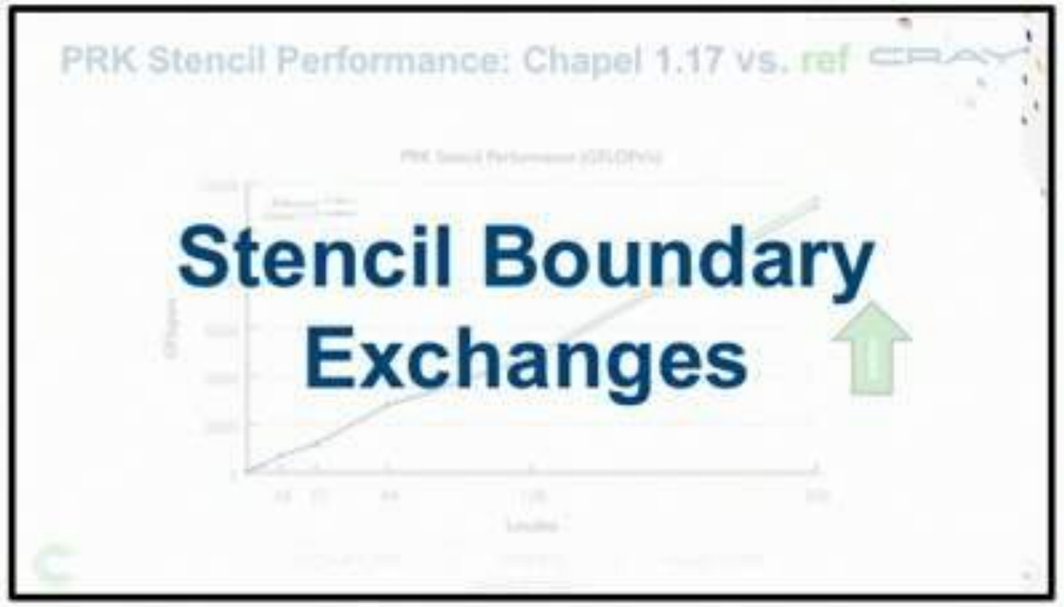
HPCC RA


STREAM Triad

ISx

PRK Stencil






Nightly performance tickers online at:
https://chapel-lang.org/perf-nightly.html

# HPCC Random Access Kernel: MPI

```c
/* Perform updates to main table.  The scalar equivalent is:
 *
 *   for (i=0; i<NUPDATE; i++) {
 *     Ran = (Ran << 1) ^ (((s64Int) Ran < 0) ? POLY : 0);
 *     Table[Ran & (TABSIZE-1)] ^= Ran;
 *   }
 */

MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
          MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
 while (i < SendCnt) {
   /* receive messages */
   do {
     MPI_Test(&inreq, &have_done, &status);
     if (have_done) {
       if (status.MPI_TAG == UPDATE_TAG) {
         MPI_Get_count(&status, tparams.dtype64, &recvUpdates);
         bufferBase = 0;
         for (j=0; j < recvUpdates; j ++) {
           inmsg = LocalRecvBuffer[bufferBase+j];
           LocalOffset = (inmsg & (tparams.TableSize - 1)) -
                         tparams.GlobalStartMyProc;
           HPCC_Table[LocalOffset] ^= inmsg;
         }
       } else if (status.MPI_TAG == FINISHED_TAG) {
         NumberReceiving--;
       } else
         MPI_Abort( MPI_COMM_WORLD, -1 );
       MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
                 MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
     }
   } while (have_done && NumberReceiving > 0);
   if (pendingUpdates < maxPendingUpdates) {
     Ran = (Ran << 1) ^ ((s64Int) Ran < ZERO64B ? POLY : ZERO64B);
     GlobalOffset = Ran & (tparams.TableSize-1);
     if ( GlobalOffset < tparams.Top)
       WhichPe = ( GlobalOffset / (tparams.MinLocalTableSize + 1) );
     else
       WhichPe = ( (GlobalOffset - tparams.Remainder) /
                   tparams.MinLocalTableSize );
     if (WhichPe == tparams.MyProc) {
       LocalOffset = (Ran & (tparams.TableSize - 1)) -
                     tparams.GlobalStartMyProc;
       HPCC_Table[LocalOffset] ^= Ran;
     } else {
       HPCC_InsertUpdate(Ran, WhichPe, Buckets);
       pendingUpdates++;
     }
     i++;
   }
   else {
     MPI_Test(&outreq, &have_done, MPI_STATUS_IGNORE);
     if (have_done) {
       outreq = MPI_REQUEST_NULL;
       pe = HPCC_GetUpdates(Buckets, LocalSendBuffer, localBufferSize,
                            &peUpdates);
       MPI_Isend(&LocalSendBuffer, peUpdates, tparams.dtype64, (int)pe,
                 UPDATE_TAG, MPI_COMM_WORLD, &outreq);
       pendingUpdates -= peUpdates;
     }
   }
 }
/* send remaining updates in buckets */
while (pendingUpdates > 0) {
  /* receive messages */
  do {
    MPI_Test(&inreq, &have_done, &status);
    if (have_done) {
      if (status.MPI_TAG == UPDATE_TAG) {
        MPI_Get_count(&status, tparams.dtype64, &recvUpdates);
        bufferBase = 0;
        for (j=0; j < recvUpdates; j ++) {
          inmsg = LocalRecvBuffer[bufferBase+j];
          LocalOffset = (inmsg & (tparams.TableSize - 1)) -
                        tparams.GlobalStartMyProc;
          HPCC_Table[LocalOffset] ^= inmsg;
        }
      } else if (status.MPI_TAG == FINISHED_TAG) {
        /* we got a done message.  Thanks for playing... */
        NumberReceiving--;
      } else {
        MPI_Abort( MPI_COMM_WORLD, -1 );
      }
      MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
                MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
    }
  } while (have_done && NumberReceiving > 0);

  MPI_Test(&outreq, &have_done, MPI_STATUS_IGNORE);
  if (have_done) {
    outreq = MPI_REQUEST_NULL;
    pe = HPCC_GetUpdates(Buckets, LocalSendBuffer, localBufferSize,
                         &peUpdates);
    MPI_Isend(&LocalSendBuffer, peUpdates, tparams.dtype64, (int)pe,
              UPDATE_TAG, MPI_COMM_WORLD, &outreq);
    pendingUpdates -= peUpdates;
  }
}
/* send our done messages */
for (proc_count = 0 ; proc_count < tparams.NumProcs ; ++proc_count) {
  if (proc_count == tparams.MyProc) { tparams.finish_req[tparams.MyProc] =
                                      MPI_REQUEST_NULL; continue; }
  /* send garbage - who cares, no one will look at it */
  MPI_Isend(&Ran, 0, tparams.dtype64, proc_count, FINISHED_TAG,
            MPI_COMM_WORLD, tparams.finish_req + proc_count);
}
/* Finish everyone else up... */
while (NumberReceiving > 0) {
  MPI_Wait(&inreq, &status);
  if (status.MPI_TAG == UPDATE_TAG) {
    MPI_Get_count(&status, tparams.dtype64, &recvUpdates);
    bufferBase = 0;
    for (j=0; j < recvUpdates; j ++) {
      inmsg = LocalRecvBuffer[bufferBase+j];
      LocalOffset = (inmsg & (tparams.TableSize - 1)) -
                    tparams.GlobalStartMyProc;
      HPCC_Table[LocalOffset] ^= inmsg;
    }
  } else if (status.MPI_TAG == FINISHED_TAG) {
    /* we got a done message.  Thanks for playing... */
    NumberReceiving--;
  } else {
    MPI_Abort( MPI_COMM_WORLD, -1 );
  }
  MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
            MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
}

MPI_Waitall( tparams.NumProcs, tparams.finish_req, tparams.finish_statuses);
```

# HPCC Random Access Kernel: MPI

```
/* Perform updates to main table.  The scalar equivalent is:
 *
 *     for (i=0; i<NUPDATE; i++) {
 *       Ran = (Ran << 1) ^ (((s64Int) Ran < 0) ? POLY : 0);
 *       Table[Ran & (TABSIZE-1)] ^= Ran;
 *     }
 */
```

COMPUTE | STORE | ANALYZE

# HPCC Random Access Kernel: MPI

```
/* Perform updates to main table.  The scalar equivalent is:
 *
 *     for (i=0; i<NUPDATE; i++) {
 *       Ran = (Ran << 1) ^ (((s64Int) Ran < 0) ? POLY : 0);
 *       Table[Ran & (TABSIZE-1)] ^= Ran;
 *     }
 */
```

**MPI Comment**


```
/* Perform updates to main table. The scalar equivalent is:
 *
 *     for (i=0; i<NUPDATE; i++) {
 *       Ran = (Ran << 1) ^ (((s64Int) Ran < 0) ? POLY : 0);
 *       Table[Ran & (TABSIZE-1)] ^= Ran;
 *     }
 */
```

# HPCC Random Access Kernel: MPI


```
/* Perform updates to main table.  The scalar equivalent is:
 *
 *     for (i=0; i<NUPDATE; i++) {
 *       Ran = (Ran << 1) ^ (((s64Int) Ran < 0) ? POLY : 0);
 *       Table[Ran & (TABSIZE-1)] ^= Ran;
 *     }
 */
```

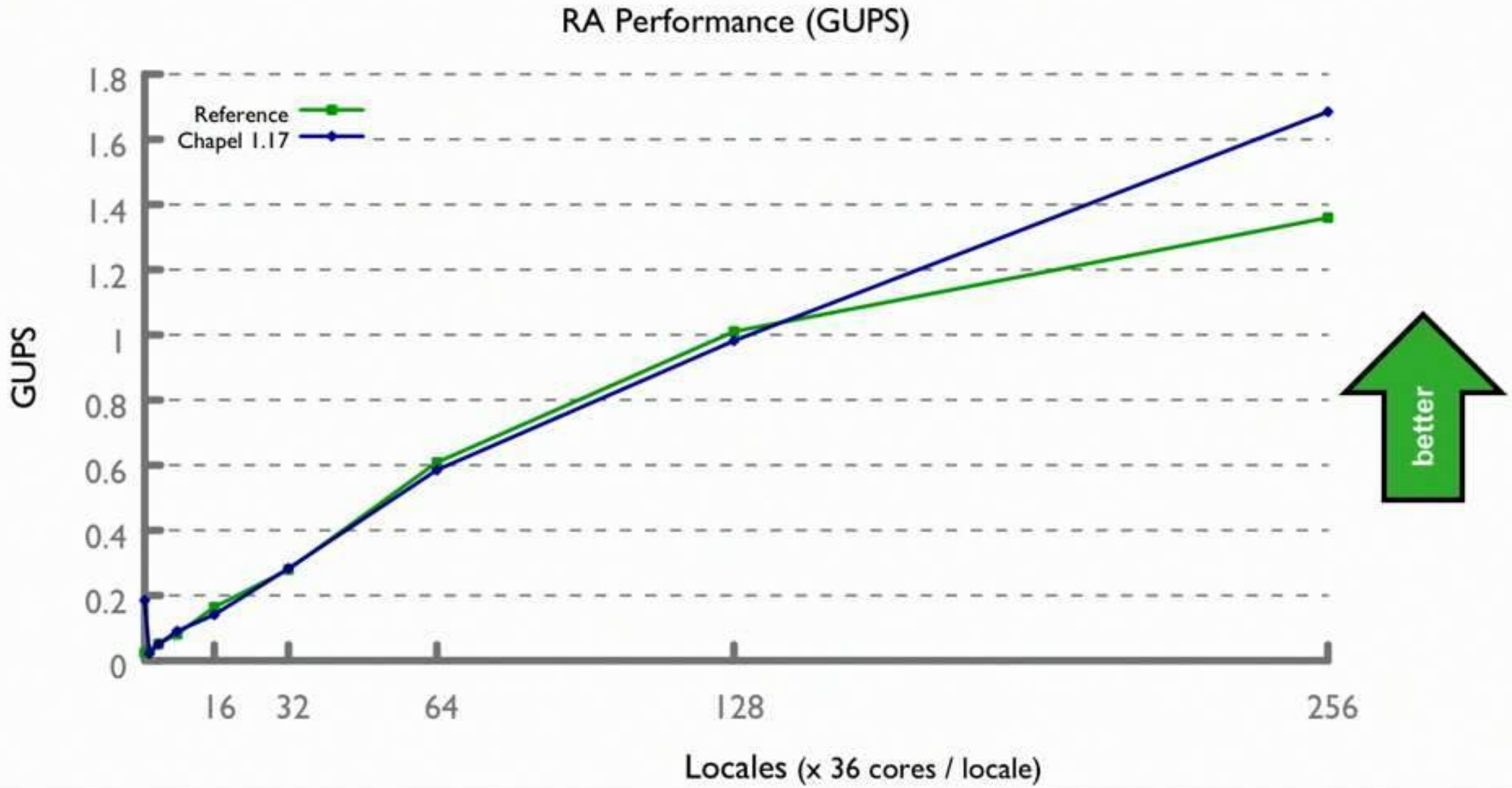

**Chapel Kernel**

```
forall (_, r) in zip(Updates, RAStream()) do
  T[r & indexMask] ^= r;
```

**MPI Comment**

```
/* Perform updates to main table. The scalar equivalent is:
 *
 *     for (i=0; i<NUPDATE; i++) {
 *       Ran = (Ran << 1) ^ (((s64Int) Ran < 0) ? POLY : 0);
 *       Table[Ran & (TABSIZE-1)] ^= Ran;
 *     }
 */
```

# RA Performance: Chapel Now vs. reference



# What's Next?

**CHIUW 2018:** The 5th annual Chapel Implementers and Users Workshop

- Vancouver BC, Friday May 25th
- Details: https://chapel-lang.org/CHIUW2018.html

# What's Next?

**CHIUW 2018:** The 5th annual Chapel Implementers and Users Workshop

- Vancouver BC, Friday May 25th
- Details: https://chapel-lang.org/CHIUW2018.html

**Chapel's college years:** plans for 2018-2021

- Further Performance and Scalability Improvements
- Libfabric/OFI Support
- GPU Support
- Cloud Support
- Chapel AI

# The Chapel Team at Cray (May 2018)

13 full-time employees + ~2 summer interns

# Chapel Community Partners

HAVERFORD COLLEGE

AMD

THE GEORGE WASHINGTON UNIVERSITY WASHINGTON, DC

WESTERN WASHINGTON UNIVERSITY

東京大学 THE UNIVERSITY OF TOKYO

THE UNIVERSITY OF ARIZONA

RICE

UNIVERSITY OF MARYLAND

BERKELEY LAB
Lawrence Berkeley National Laboratory

Lawrence Livermore National Laboratory

Sandia National Laboratories

DEPARTMENT OF DEFENSE UNITED STATES OF AMERICA

Yale

(and several others…)

https://chapel-lang.org/collaborations.html

# Summary

- **Chapel's made huge progress over the past five years**
- **Ready for use in production***
- **Open to collaborations**
  - Plenty of research questions remain

# Musical Ornaments

John Leo

Halfaya Research

May 14, 2018

# Outline

- A look toward the future
- Music Tools
- Equivalences and Ornaments

Sources:

- `https://github.com/halfaya/MusicTools`

# Robert Harper

*Eventually all the arbitrary programming languages are going to be just swept away with the oceans, and we will have the permanence of constructive, intuistionistic type theory as the master theory of computation—without doubt, in my mind, no question. So, from my point of view—this is a personal statement—working in anything else is a waste of time.*

CMU Homotopy Type Theory lecture 1, 52:56–53:20.

# What will programming look like in 50 years?

- Convergence of math and computer science
- Functional Programming, Algebra of Programming
- Dependent Types or a successor (Cubical?)
- Who does the programming?

# How do we get there from here?

- Add dependent types to an industrial-strength language (Haskell)
- Make a dependently typed language (Agda, Idris) practical to use
- Learn how to program using dependent types
- Many theoretical and practical advances are still needed

# Euterpea

*The Haskell School of Music*

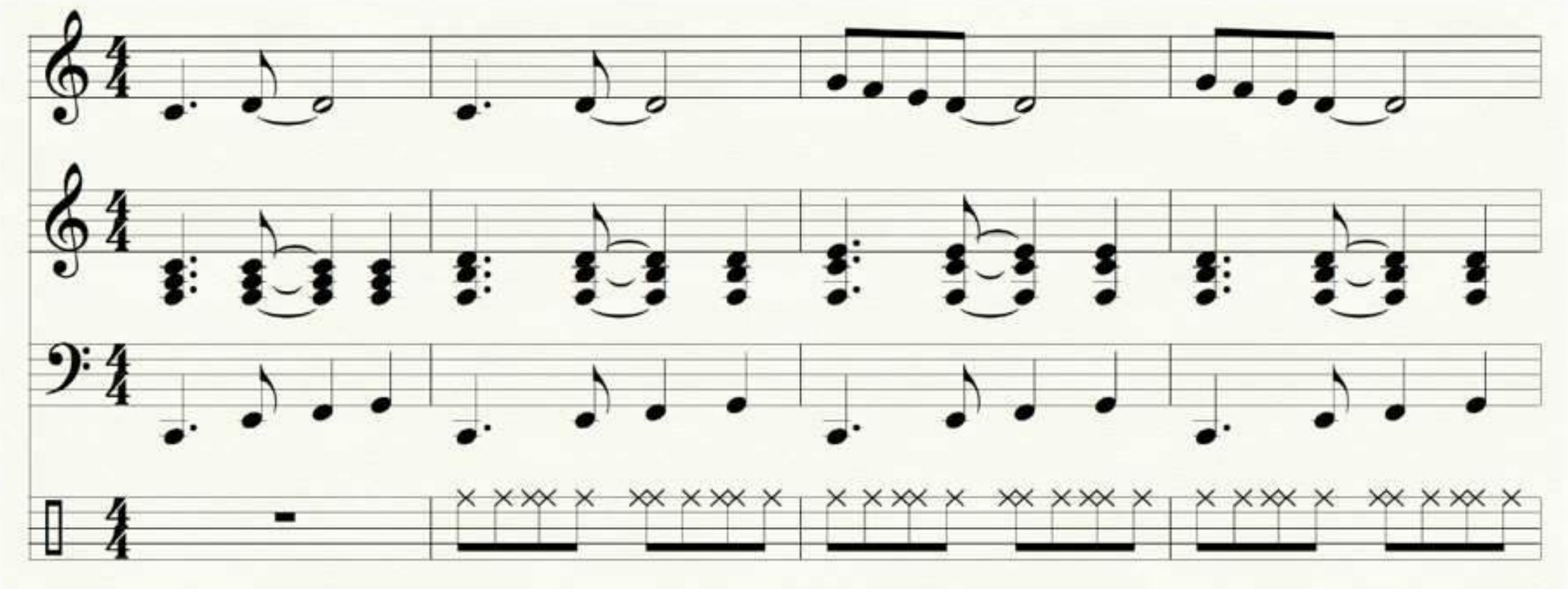
— From Signals to Symphonies —

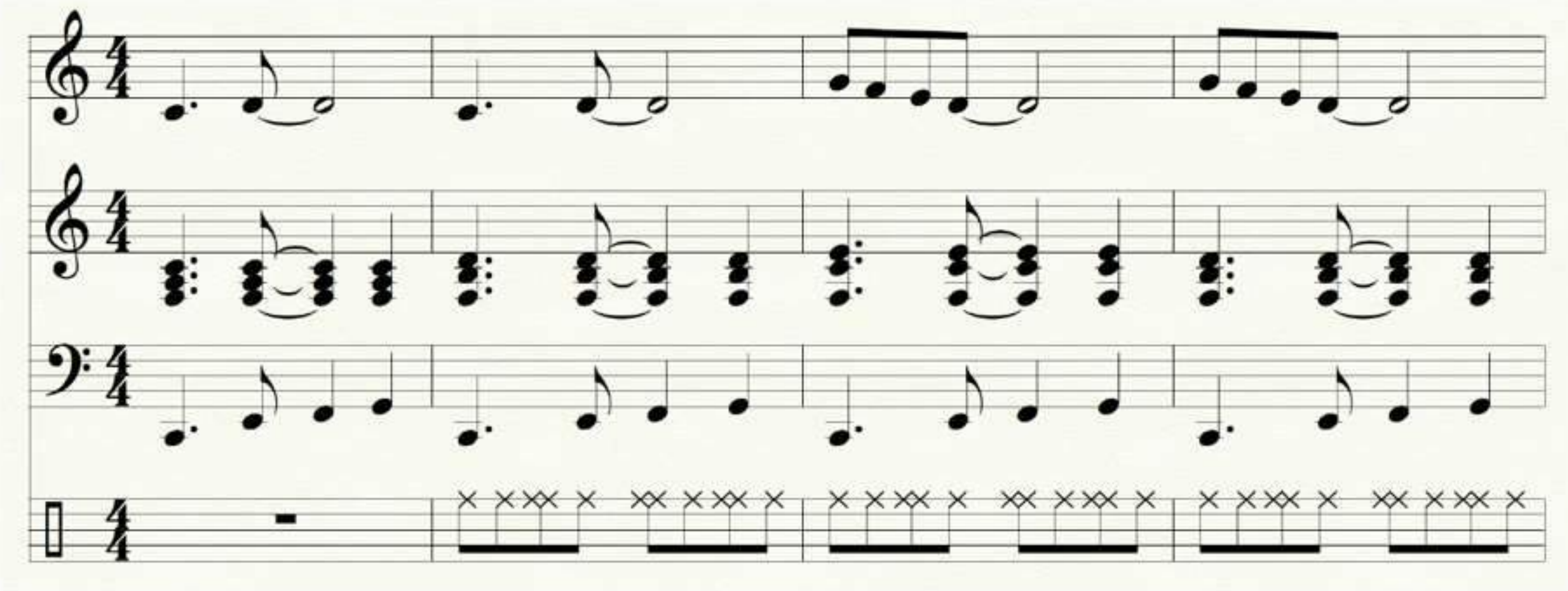
**Paul Hudak**

Yale University
Department of Computer Science

# Music Tools

- Collection of composable tools for synthesis and analysis of music
- Originally written in Haskell
- Converted to Agda, using Haskell for MIDI interface
- Explore programming using dependent types in a circumscribed yet rich domain
- Use math, including transport of equivalences (from HoTT) and Ornaments

# Look vs Time (1997)

Logic Pro X
File
Edit
Track
Navigate
Record
Mix
View
Window
1
Help
Mon 4:27 PM
John Leo
Untitled - Tracks
84
4/4
Cmaj
Filter: All Instruments
Region: Default
Style: 0
Quantize: Default
Interpretation:
Syncopation:
No Overlap:
Max. Dots: 1
Event: Insert Defaults
MIDI Channel: 1
Velocity: 0
Text: Plain Text
Lyric:
Part Box
Edit
Functions
View
Snap:
Smart
Drag:
No Overlap
Marimba
Fingerstyle Bass
SoCal
Accomp
Bass (+12)
Drums
Piano Roll
Score
Step Editor
Layout
SoCal
Drum Kit
Channel EQ
Compressor
Stereo Out
Read
Bnce
Stereo Out

# Look vs Time (1997)

Emacs File Edit Options Buffers Tools Agda Help    86% Mon 4:27 PM John Leo

LookVsTime.agda

```
open import MidiEvent
open import Util

tempo : ℕ
tempo = 84

----

melodyChannel : Channel-1
melodyChannel = # 0

melodyInstrument : InstrumentNumber-1
melodyInstrument = # 8 -- celesta

melodyNotes : List Note
melodyNotes =
  note (8th 3) (c 3) ∷
  note (8th 5) (d 3) ∷

  note (8th 3) (c 3) ∷
  note (8th 5) (d 3) ∷

  note (8th 1) (g 3) ∷
  note (8th 1) (f 3) ∷
  note (8th 1) (e 3) ∷
  note (8th 5) (d 3) ∷

  note (8th 1) (g 3) ∷
  note (8th 1) (f 3) ∷
  note (8th 1) (e 3) ∷
  note (8th 5) (d 3) ∷

  note (8th 1) (a 3) ∷
```

```
U:---  LookVsTime.agda   8% L32   Git:master  (Agda:Checked +2)
```

# Music Representation à la Euterpea

```
data Pitch : Set where
  pitch : ℤ → Pitch

data Duration : Set where
  duration : ℕ → Duration

data Note : Set where
  note : Duration → Pitch → Note
  rest : Duration → Note

data Music : Set where
  note : Note → Music
  _::_ : Music → Music → Music -- sequential
  _||_ : Music → Music → Music -- parallel
```

# Equivalent Representations of Pitch

```
data Pitch : Set where
  pitch : ℤ → Pitch

chromaticScaleSize : ℕ
chromaticScaleSize = 12

data RelativePitch : Set where
  relativePitch : Fin chromaticScaleSize → RelativePitch

data Octave : Set where
  octave : ℤ → Octave

PitchOctave : Set
PitchOctave = RelativePitch × Octave
```

# Equivalences

- Define an equivalence between `Pitch` and `PitchOctave`
- Using HoTT techniques, automatically lift this equivalence to functions defined using `Pitch`
- See *Equivalences for Free!* (Tabareau, Tanter, Sozeau)
- Challenge: Defining base equivalences. Can this be automated?

# Ornaments

```
data Music a  = ...
  |  Modify Control (Music a)

data Control = ...
  |  Phrase [PhraseAttribute]

data PhraseAttribute = ...
  |  Orn Ornament

data Ornament =
  Trill | Mordent | InvMordent | DoubleMordent |
  Turn | TrilledTurn | ShortTrill ...
```

# Ornaments

- Functions on a base `Music` structure can be automatically lifted to operate on `Music` ornamented with additional information
- See works on Ornaments by McBride, Dagand and others
- Challenge: Shallow embedding of ornaments in Agda

# Conclusion

- Music is a good domain in which to explore practical application of dependent types
- Using math can be more work at first, but should be a big win in the long term
- Figure out how to minimize the work and maximize the reward