Legion Programming System

BY BLAKE COLLINS

WITH SOME SLIDES USED BY ELLIOT SLAUGHTER, WONCHAN LEE, TODD WARSZAWSKI, AND KARTHIK MURTHY

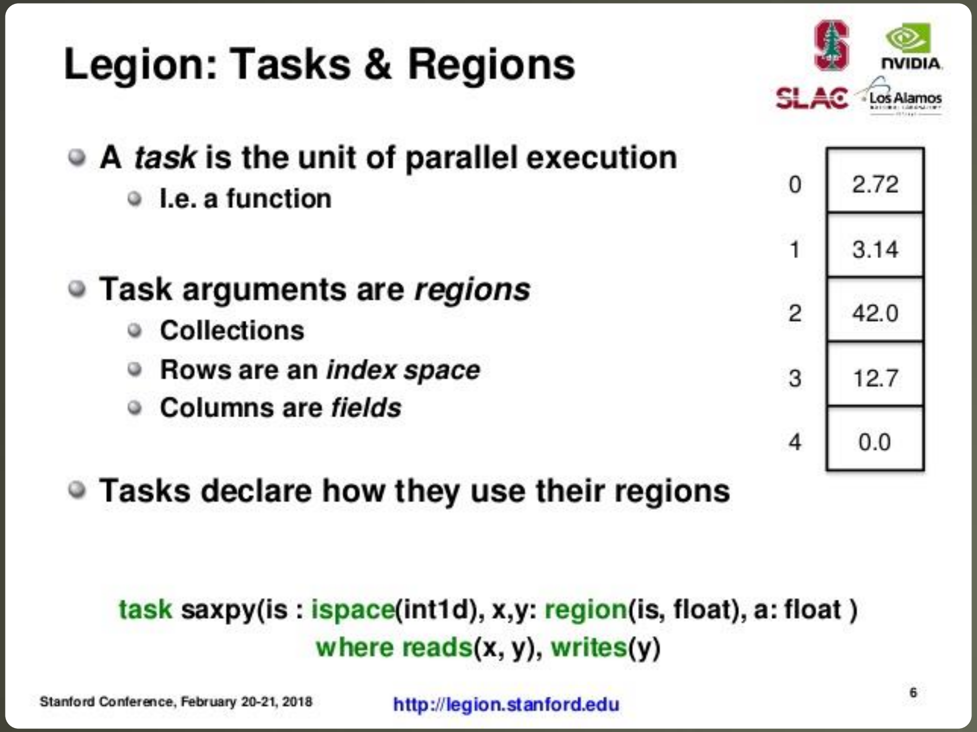
What is it?

Legion is a data-centric programming model for writing high-performance applications for distributed heterogeneous architectures. It holds several advantages such as it's ability to handle data movement and parallelism, as well as allow programmers to explicitly declare properties of program data including organization, partitioning, privileges, and coherence.

Legion also provides a mapping interface which gives programmers direct control over all the details of how an application is mapped and executed. Furthermore, Legion's understanding of program data makes the mapping process orthogonal to correctness. This simplifies program performance tuning and enables easy porting of applications to new architectures.

How does it work?

- It handles data to increase the efficiency of programs constructed with it. It does this partitioning the data into logical regions and then being designated tasks that operate on those given regions, creating further subroutines as needed. Each region is described by an index space of rows (either unstructured pointers or structured 1D, 2D, or 3D arrays) and a field space of columns.
- Mapping decisions regarding how tasks are assigned to processors and how physical instances of logical regions are assigned to memories are made entirely by mappers. Mappers are part of application code and implement a mapping interface. Mappers are queried by the Legion runtime whenever any mapping decision needs to be made. Legion guarantees that mapping decisions only impact performance and are orthogonal to correctness which simplifies tuning of Legion applications and enables easy porting to different architectures.

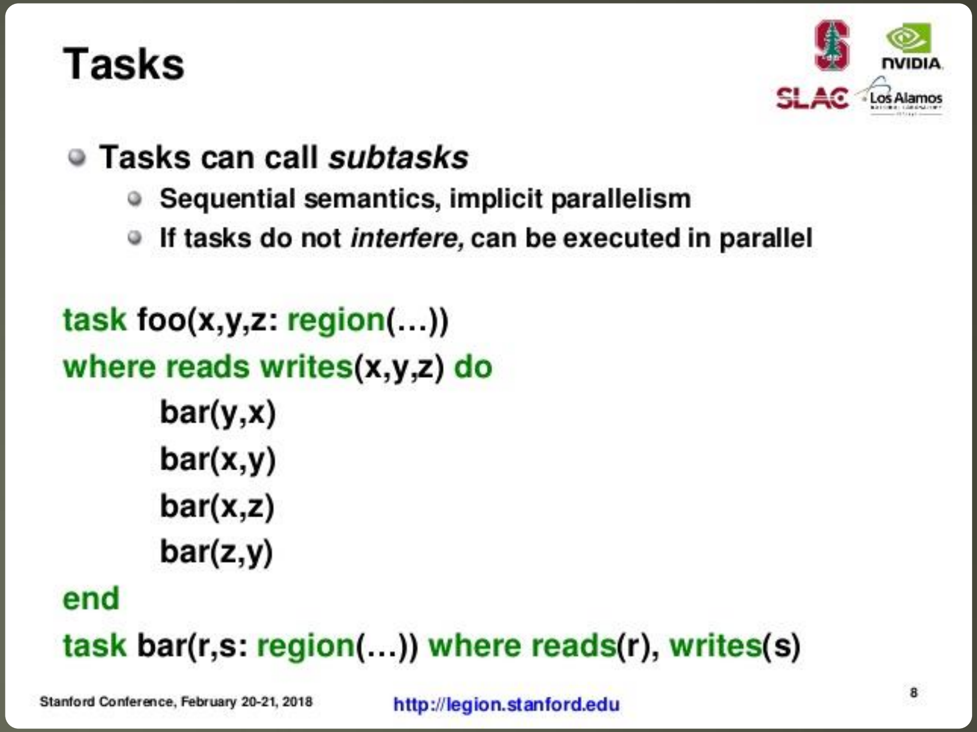


Legion: Tasks & Regions

- A *task* is the unit of parallel execution
  - I.e. a function
- Task arguments are *regions*
  - Collections
  - Rows are an *index space*
  - Columns are *fields*
- Tasks declare how they use their regions

0	2.72
1	3.14
2	42.0
3	12.7
4	0.0

```
task saxpy(is : ispace(int1d), x,y: region(is, float), a: float )
        where reads(x, y), writes(y)
```

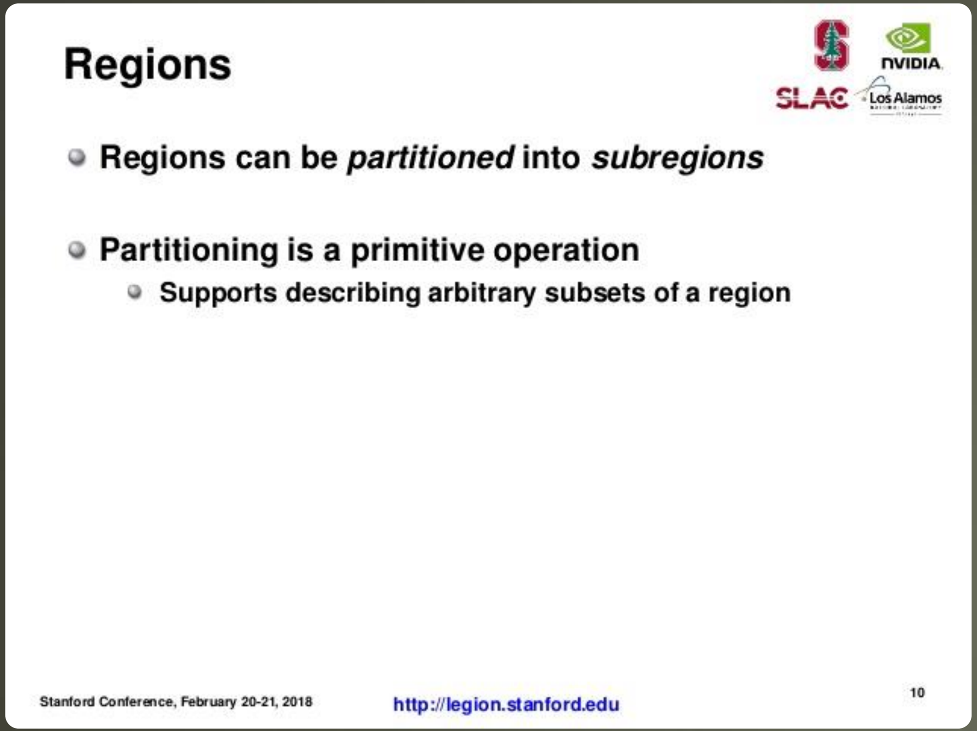


Tasks

- Tasks can call *subtasks*
  - Sequential semantics, implicit parallelism
  - If tasks do not *interfere*, can be executed in parallel

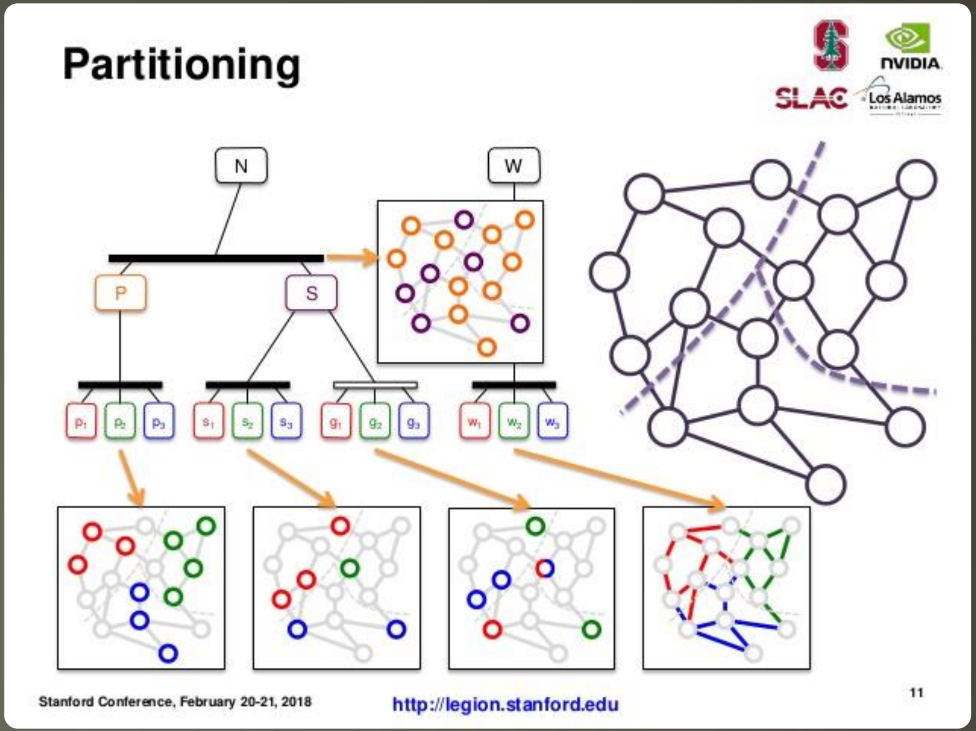
```
task foo(x,y,z: region(…))
where reads writes(x,y,z) do
        bar(y,x)
        bar(x,y)
        bar(x,z)
        bar(z,y)
end
task bar(r,s: region(…)) where reads(r), writes(s)
```

Example Slides 1



Regions

- Regions can be *partitioned* into *subregions*
- Partitioning is a primitive operation
  - Supports describing arbitrary subsets of a region



Partitioning

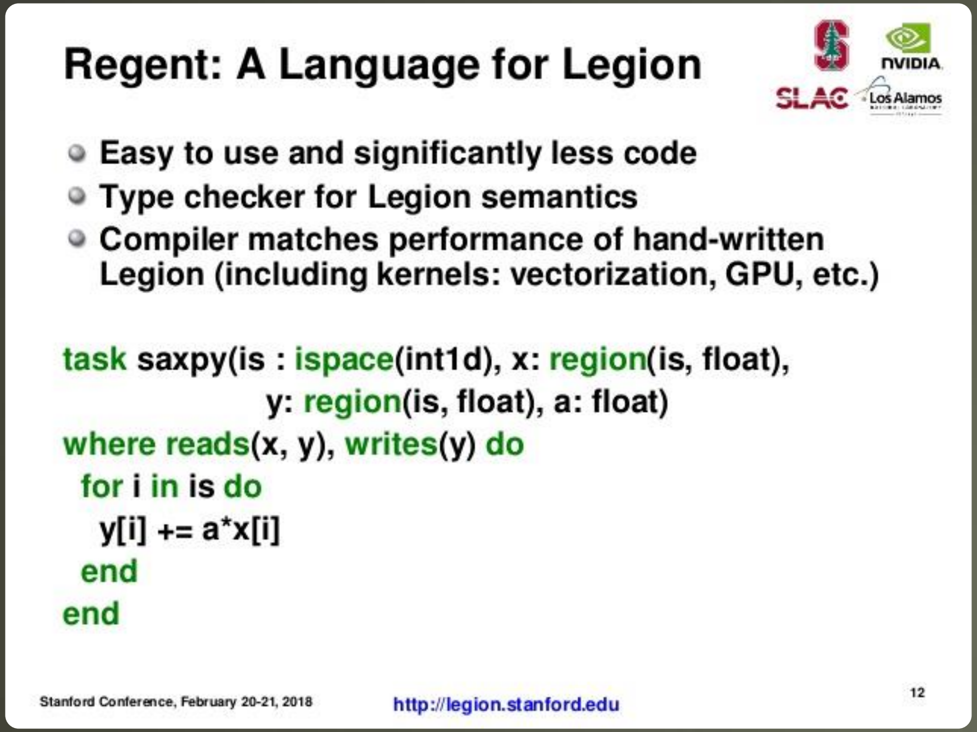
Example Slides 2

Who Uses It? How popular is it?

- Advanced application developers: programmers who traditionally have used combinations of MPI, GASNet, Pthreads, OpenCL, and/or CUDA to develop their applications and always re-write applications from scratch for maximum performance on each new architecture.
- Domain specific language and library authors: tool writers who develop high-level productivity languages and libraries that support separate implementations for every target architecture for maximum performance.
- Unfortunately, I could not evaluate how popular the program is directly but looking through the list of companies that fund them, two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration), I can assume that probably indicates that they use them for the purposes of their programs.

Use or Relation to other HPC languages? Why is it useful in conjunction with other HPC languages?

- As stated earlier, Legion Programming System is for users who use a combination of other HPC languages, such as MPI, OpenCL, and others, for the purposes of rewriting code from scratch to maximize performance for each new architecture.
- It interfaces with them and can be programmed to understand them, and establishes a memory hierarchy for them, allowing them to run more efficiently by automating data flow. This program was made to work in tandem with other HPC languages to increase their efficiency.

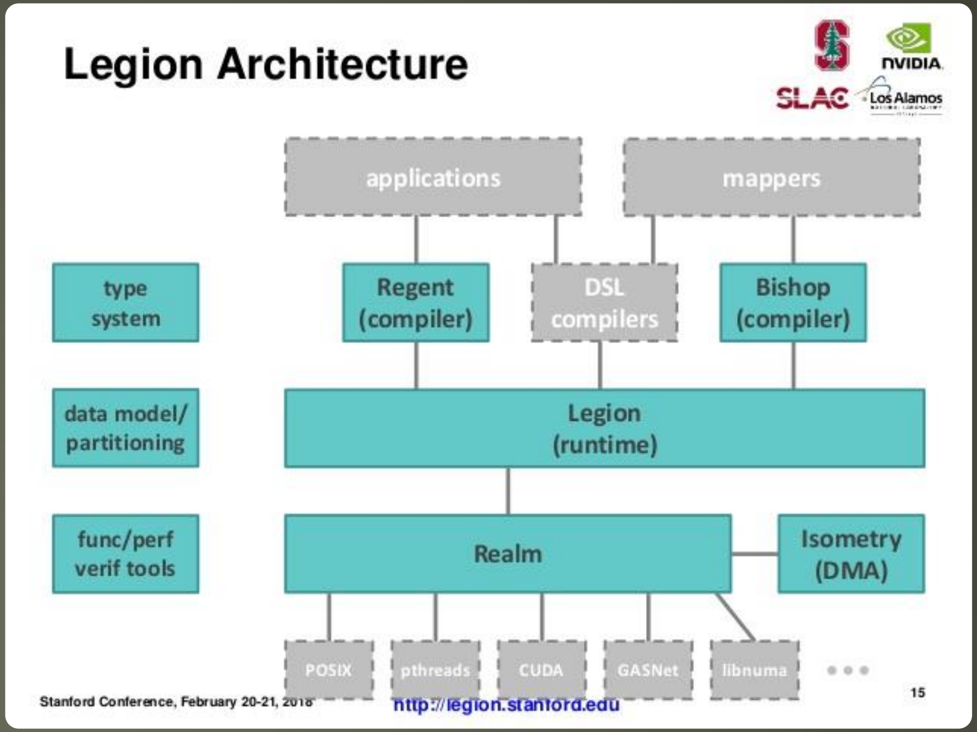


Regent: A Language for Legion

- Easy to use and significantly less code
- Type checker for Legion semantics
- Compiler matches performance of hand-written Legion (including kernels: vectorization, GPU, etc.)

```
task saxpy(is : ispace(int1d), x: region(is, float),
                       y: region(is, float), a: float)
where reads(x, y), writes(y) do
  for i in is do
    y[i] += a*x[i]
  end
end
```

Stanford Conference, February 20-21, 2018 http://legion.stanford.edu 12



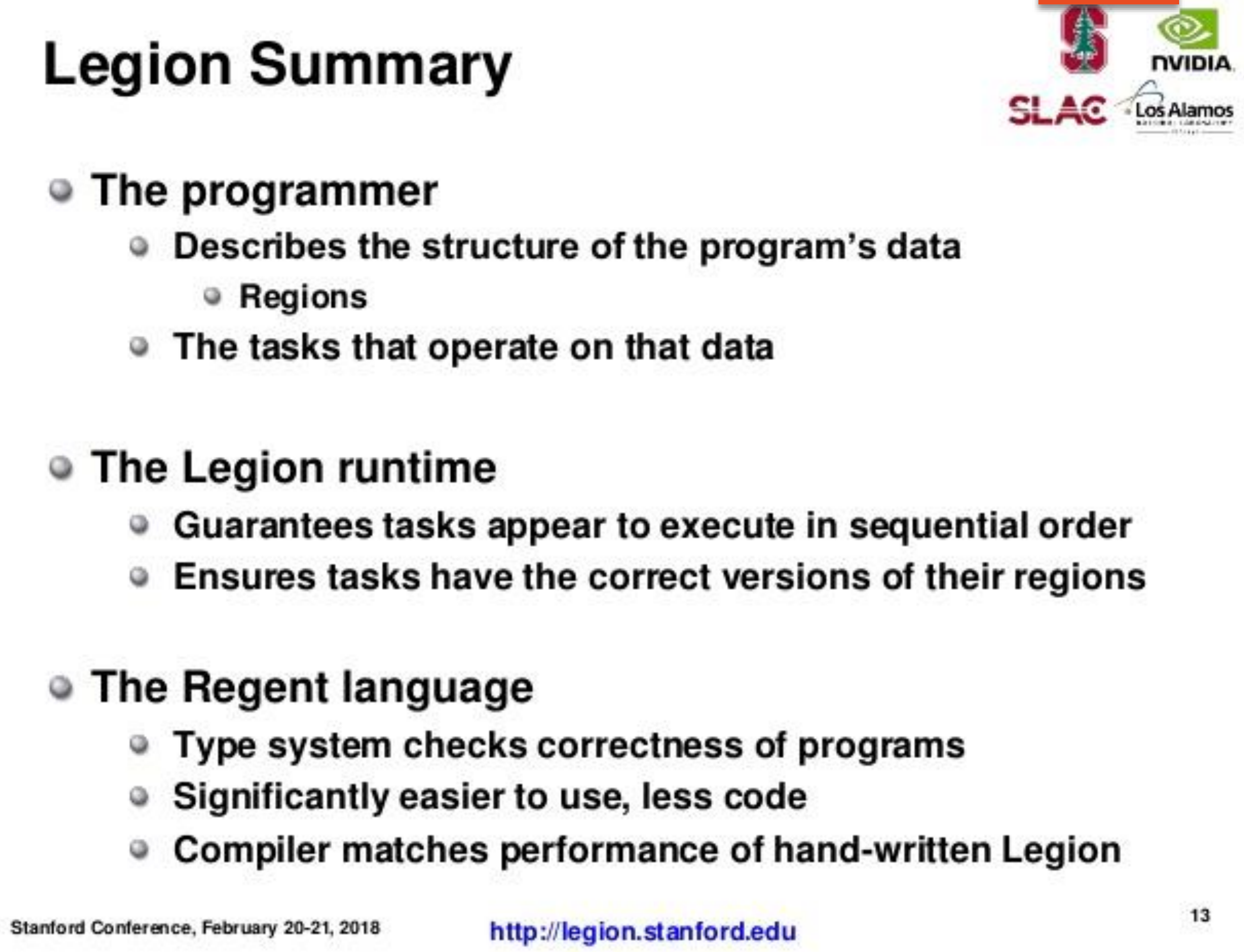
Legion Architecture

Stanford Conference, February 20-21, 2018 http://legion.stanford.edu 15

Nonrequired mentions & Example Slides 3

Closing Summary

- Legion Programming System is a software package dedicated to working in tandem with other HPC languages.
- It's capable of handling data flow and ensuring that all tasks execute as directed in each region by the programmer.



Legion Summary

- The programmer
  - Describes the structure of the program's data
    - Regions
  - The tasks that operate on that data
- The Legion runtime
  - Guarantees tasks appear to execute in sequential order
  - Ensures tasks have the correct versions of their regions
- The Regent language
  - Type system checks correctness of programs
  - Significantly easier to use, less code
  - Compiler matches performance of hand-written Legion

Stanford Conference, February 20-21, 2018 http://legion.stanford.edu 13

Websites used

- https://legion.stanford.edu/overview/ for the information about Legion.

- https://www.slideshare.net/insideHPC/advances-in-the-legion-programming-model for additional slides used in my presentation, the authors of which were mentioned at the beginning of the presentation.