# Tasks all the way down

## Parallelism in julia

Keno Fischer
Co-Founder & CTO

# Rapid adoption for a young language

20M downloads; 4,000 packages; 10,000 companies; 1,500 universities

## IEEE Spectrum Language Rankings

**Choose a Ranking**

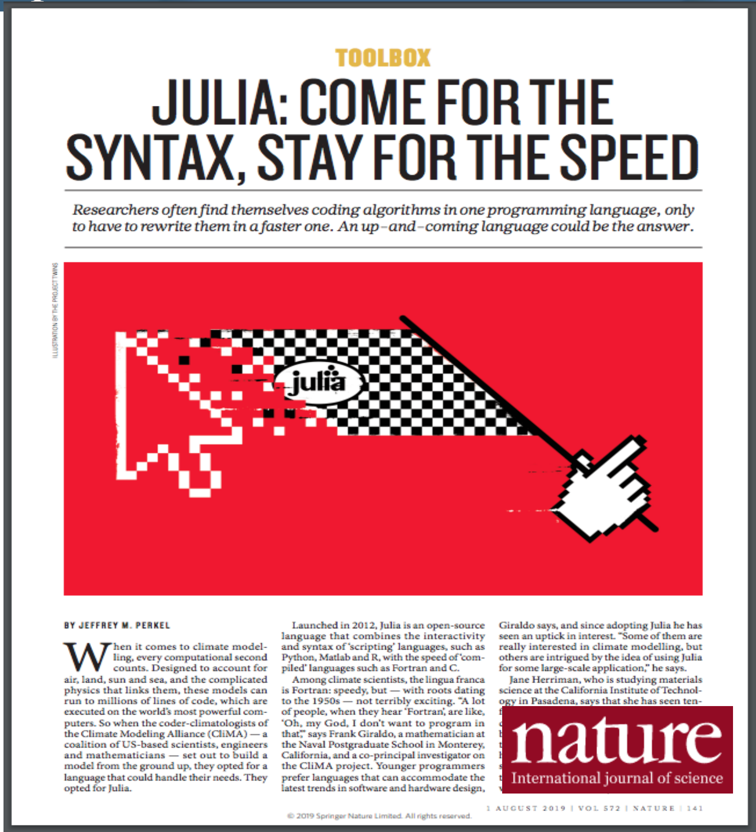
IEEE Spectrum | Trending | Jobs | Open | Custom

**Language Types**

Web | Enterprise | Mobile | Embedded

(Click to hide)

| | | | |
|---|---|---|---|
| 17 | HTML | Web | 61.4 |
| 18 | Kotlin | Web, Mobile | 57.8 |
| 19 | Julia | Enterprise | 56.0 |
| 20 | Rust | Web, Enterprise, Embedded | 55.6 |
| 21 | Shell | Enterprise | 52.0 |
| 22 | Processing | Web, Enterprise | 49.2 |



TOOLBOX

JULIA: COME FOR THE SYNTAX, STAY FOR THE SPEED

*Researchers often find themselves coding algorithms in one programming language, only to have to rewrite them in a faster one. An up-and-coming language could be the answer.*

BY JEFFREY M. PERKEL

When it comes to climate modelling, every computational second counts. Designed to account for air, land, sun and sea, and the complicated physics that links them, these models can run to millions of lines of code, which are executed on the world's most powerful computers. So when the coder-climatologists of the Climate Modeling Alliance (CliMA) — a coalition of US-based scientists, engineers and mathematicians — set out to build a model from the ground up, they opted for a language that could handle their needs. They opted for Julia.

Launched in 2012, Julia is an open-source language that combines the interactivity and syntax of 'scripting' languages, such as Python, Matlab and R, with the speed of 'compiled' languages such as Fortran and C.

Among climate scientists, the lingua franca is Fortran: speedy, but — with roots dating to the 1950s — not terribly exciting. "A lot of people, when they hear 'Fortran', are like, 'Oh, my God, I don't want to program in that'," says Frank Giraldo, a mathematician at the Naval Postgraduate School in Monterey, California, and a co-principal investigator on the CliMA project. Younger programmers prefer languages that can accommodate the latest trends in software and hardware design,

Giraldo says, and since adopting Julia he has seen an uptick in interest. "Some of them are really interested in climate modelling, but others are intrigued by the idea of using Julia for some large-scale application," he says.

Jane Herriman, who is studying materials science at the California Institute of Technology in Pasadena, says that she has seen ten-

nature
International journal of science

# Customers, Partners & Companies Using Julia

10,000+ companies using Julia

Many case studies here:
https://juliacomputing.com/case-studies/

# Universities Using and Teaching Julia

BROWN UNIVERSITY

CORNELL UNIVERSITY FOUNDED A.D. 1865

EMORY UNIVERSITY

NORTHEASTERN UNIVERSITY BOSTON, MASSACHUSETTS

PSU PENN STATE UNIVERSITY

PURDUE

SZKOŁA GŁÓWNA HANDLOWA W WARSZAWIE

Stanford University

THE UNIVERSITY OF EDINBURGH

UNIVERSITY of GLASGOW

UNIVERSITY of SOUTH FLORIDA 1956

UNIVERSITE PAUL SABATIER TOULOUSE III

UNIVERSITY OF CONNECTICUT 1881

FIDE SPLENDET ET SCIENTIA

THE UNIVERSITY OF CALIFORNIA BERKELEY 1868

Western UNIVERSITY · CANADA

ARIZONA STATE UNIVERSITY 1885

MŰEGYETEM 1782

ILLINOIS STATE UNIVERSITY 1857

SIDERE MENS EADEM MUTATO

CUNY THE CITY UNIVERSITY OF NEW YORK

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

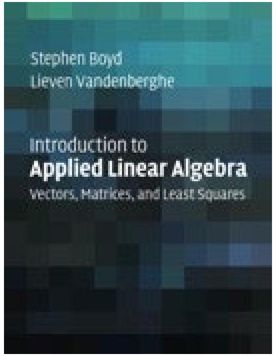
MIT

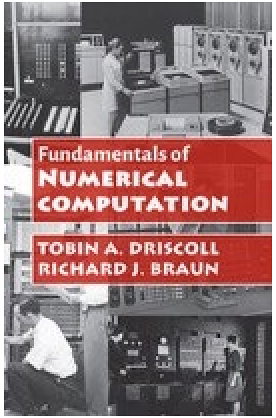
TOKYO METROPOLITAN UNIVERSITY 首都大学東京

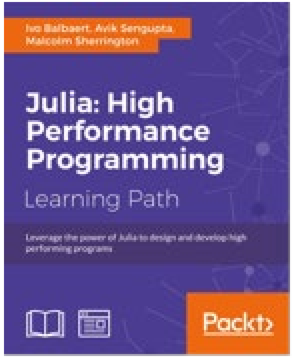
Ucla

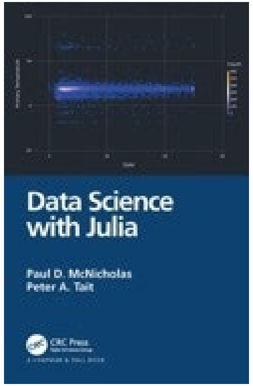
AGH

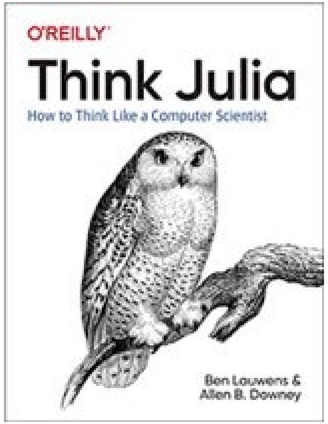
# Books on Julia

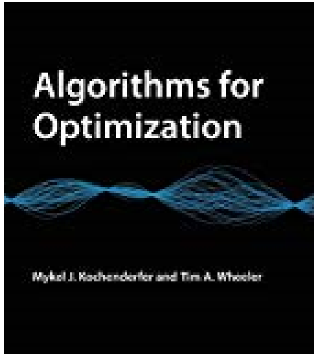


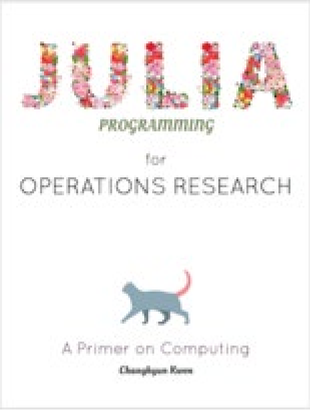



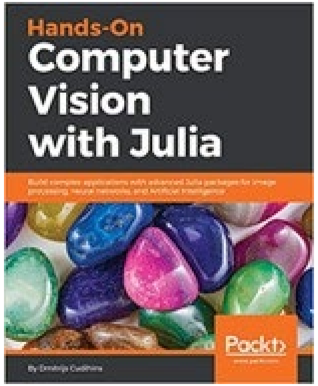




# Celeste.jl: Julia at Peta-scale

Cori: 650,000 cores, 1.3M threads, 60 TB of data

Most light sources are near the detection limit.

**Cataloging the Visible Universe through Bayesian Inference at Petascale**

Jeffrey Regier*, Kiran Pamnany†, Keno Fischer‡, Andreas Noack§, Maximilian Lam*, Jarrett Revels§, Steve Howard¶, Ryan Giordano¶, David Schlegel‖, Jon McAuliffe¶, Rollin Thomas‖, Prabhat‖

*Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
†Parallel Computing Lab, Intel Corporation
‡Julia Computing
§Computer Science and AI Laboratories, Massachusetts Institute of Technology
¶Department of Statistics, University of California, Berkeley
‖Lawrence Berkeley National Laboratory

Berkeley UNIVERSITY OF CALIFORNIA

NERSC

intel LABS

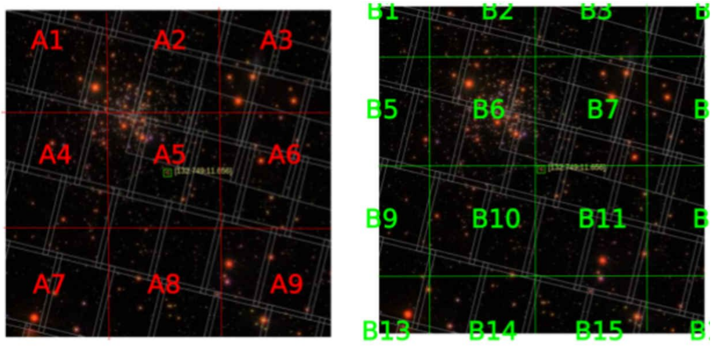
BERKELEY LAB

Julia computing

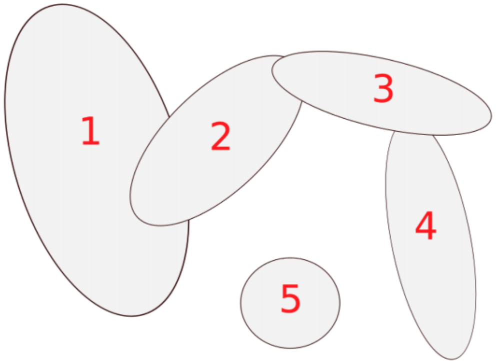
MIT

# Irregular, Multi -Scale Parallelism

## Nodes

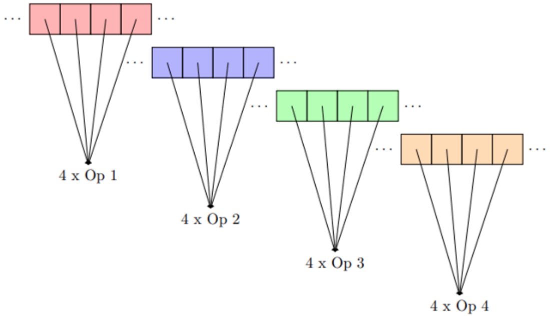



## Threads




Light sources that do not overlap may be updated concurrently.

## SIMD



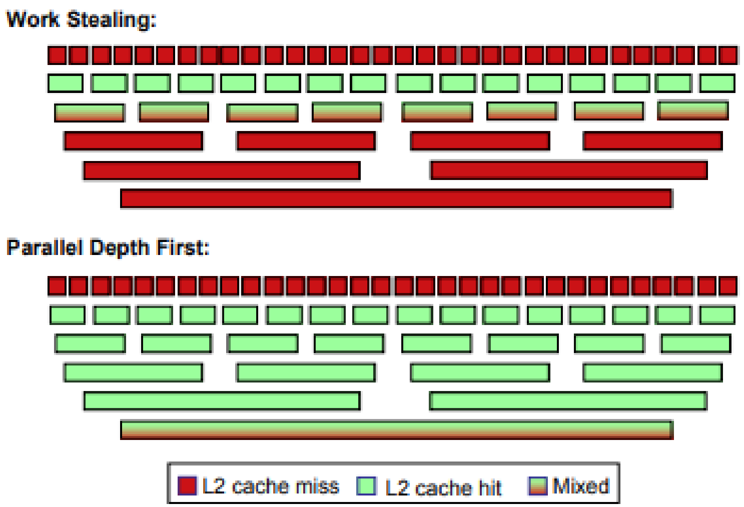
# Fundamental Schedule Unit: Task

```julia
function pfib(n::Int)
    if n <= 1
        return n
    end
    t = Threads.@spawn pfib(n-2)
    return pfib(n-1) + fetch(t)::Int
end
```

- Concurrency
  - w/ High Performance I/O Scheduler
- Parallelism
- Low memory footprint (Millions of tasks per node)
- Dynamically serializable

*Basic Threading Examples in JuliaLang v1.3 (Proceedings of JuliaCon 2019)*

# Single Node Schedule: Parallel Depth First



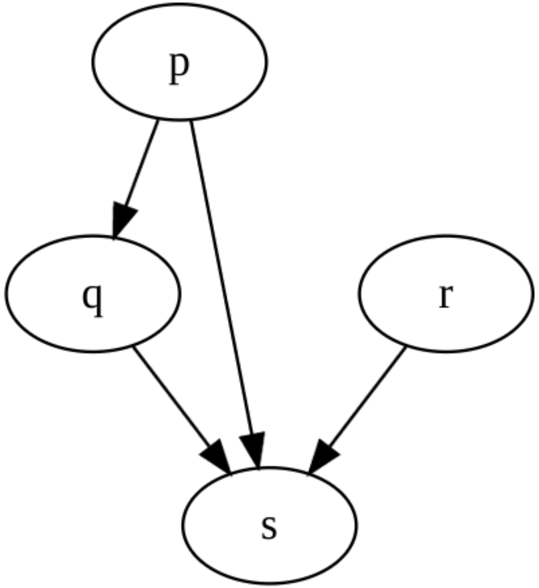

- Highly Cache efficient for regular problems
- Composability/Nested parallelism without sacrificing performance
    - Fearless parallelism for library authors

*Scheduling Threads for Constructive Cache Sharing on CMPs (SPAA '07)*

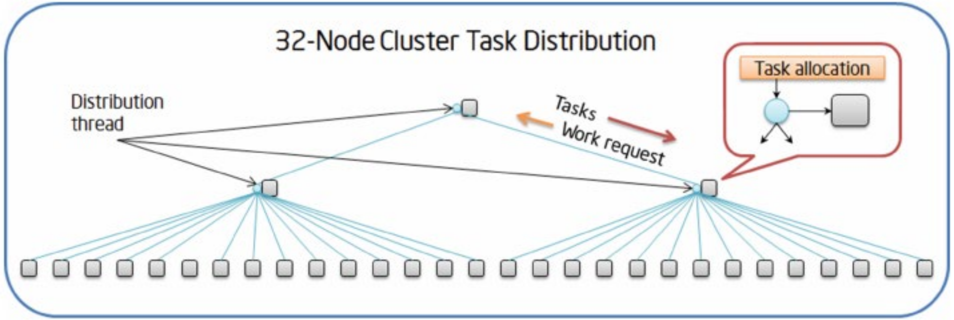
Jameson Nash　　Jeff Bezanson　　Kiran Pamnany

# Distributed Schedule: User Policy

- Different Applications need different scheduling approaches
- Common needs available from package repository




Dagger.jl: Dask-like global DAG scheduler




Gasp.jl: Dtree load-balancing irregular work scheduler (scaling to millions of concurrent threads)

# Active work

- Compiler integration
  - LLVM-level optimization of task states
  - Julia-level semantics for compiler optimized tasks
- SIMT unification (in two directions)
  - Each GPU “thread” should be a task
  - SIMT programming model for regular workloads across the abstraction levels
- Code Loading/Distributed JIT/Code Caching Opportunities