Tasks all the way down
Parallelism in julia

Keno Fischer
Co-Founder & CTO
www.juliacomputing.com
Rapid adoption for a young language

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

IEEE Spectrum Language Rankings

<table>
<thead>
<tr>
<th>Choose a Ranking</th>
<th>Language Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE Spectrum</td>
<td>Web</td>
</tr>
<tr>
<td></td>
<td>Enterprise</td>
</tr>
<tr>
<td></td>
<td>Mobile</td>
</tr>
<tr>
<td></td>
<td>Embedded</td>
</tr>
<tr>
<td>Open</td>
<td>Custom</td>
</tr>
</tbody>
</table>

Click to hide

<table>
<thead>
<tr>
<th>Rank</th>
<th>Language</th>
<th>Web</th>
<th>Mobile</th>
<th>Embed</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>HTML</td>
<td></td>
<td></td>
<td></td>
<td>61.4</td>
</tr>
<tr>
<td>18</td>
<td>Kotlin</td>
<td></td>
<td></td>
<td></td>
<td>57.8</td>
</tr>
<tr>
<td>19</td>
<td>Julia</td>
<td></td>
<td></td>
<td></td>
<td>56.0</td>
</tr>
<tr>
<td>20</td>
<td>Rust</td>
<td></td>
<td></td>
<td></td>
<td>55.6</td>
</tr>
<tr>
<td>21</td>
<td>Shell</td>
<td></td>
<td></td>
<td></td>
<td>52.0</td>
</tr>
<tr>
<td>22</td>
<td>Processing</td>
<td></td>
<td></td>
<td></td>
<td>49.2</td>
</tr>
</tbody>
</table>
Customers, Partners & Companies Using Julia

10,000+ companies using Julia

Many case studies here: https://juliacomputing.com/case-studies/
Universities Using and Teaching Julia
Books on Julia
Celeste.jl: Julia at Peta-scale

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

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

Most light sources are near the detection limit.
Irregular, Multi-Scale Parallelism

Nodes

Threads

SIMD

Light sources that do not overlap may be updated concurrently.
Fundamental Schedule Unit: Task

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)
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