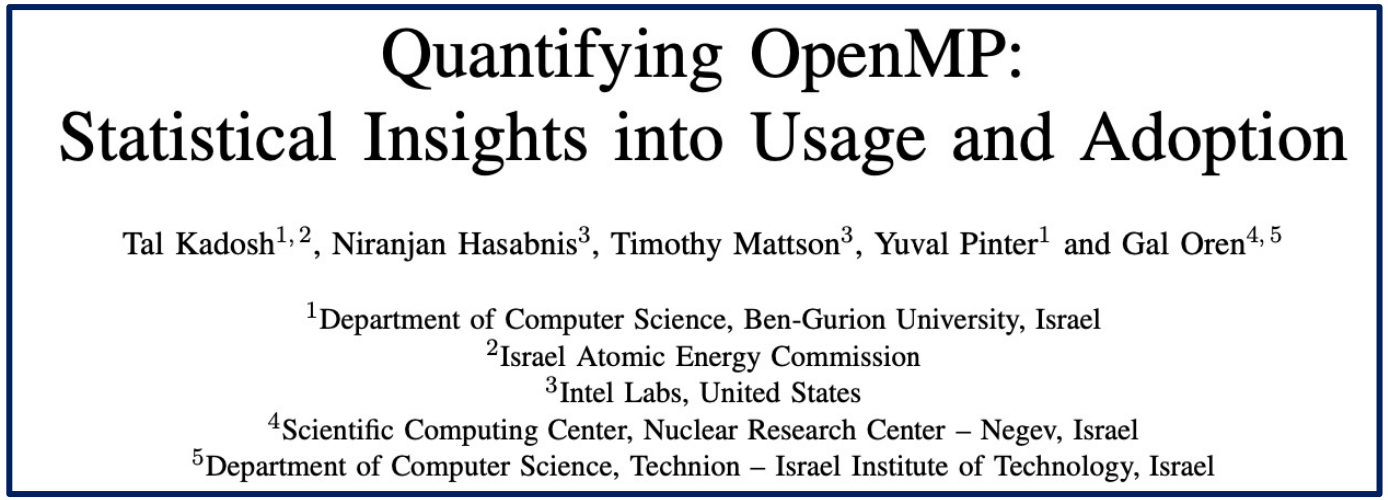
How Popular is OpenMP?

Tim Mattson

Human Learning Group*

* This is a made-up organization … so I have something to say when asked where I work.

This talk is based on this paper



Quantifying OpenMP:
Statistical Insights into Usage and Adoption

Tal Kadosh[1,2], Niranjan Hasabnis[3], Timothy Mattson[3], Yuval Pinter[1] and Gal Oren[4,5]

[1]Department of Computer Science, Ben-Gurion University, Israel
[2]Israel Atomic Energy Commission
[3]Intel Labs, United States
[4]Scientific Computing Center, Nuclear Research Center – Negev, Israel
[5]Department of Computer Science, Technion – Israel Institute of Technology, Israel

To learn more, get the paper and watch this video

Our Paper at IEEE HPEC'23. The full talk from our lead Author (Tal Kadosh): https://youtu.be/ZJxCRzPcnEw?si=VjoC8FjFJBxim[illegible]15

Download the paper here: https://arxiv.org/abs/2308.08002

The Dirty Secret of Programming Model Design

- If you talk to people who design programming models, we tend to over-estimate how much our models are used.
  - It's like the children of Lake Wobegon – *All programming models are above average*

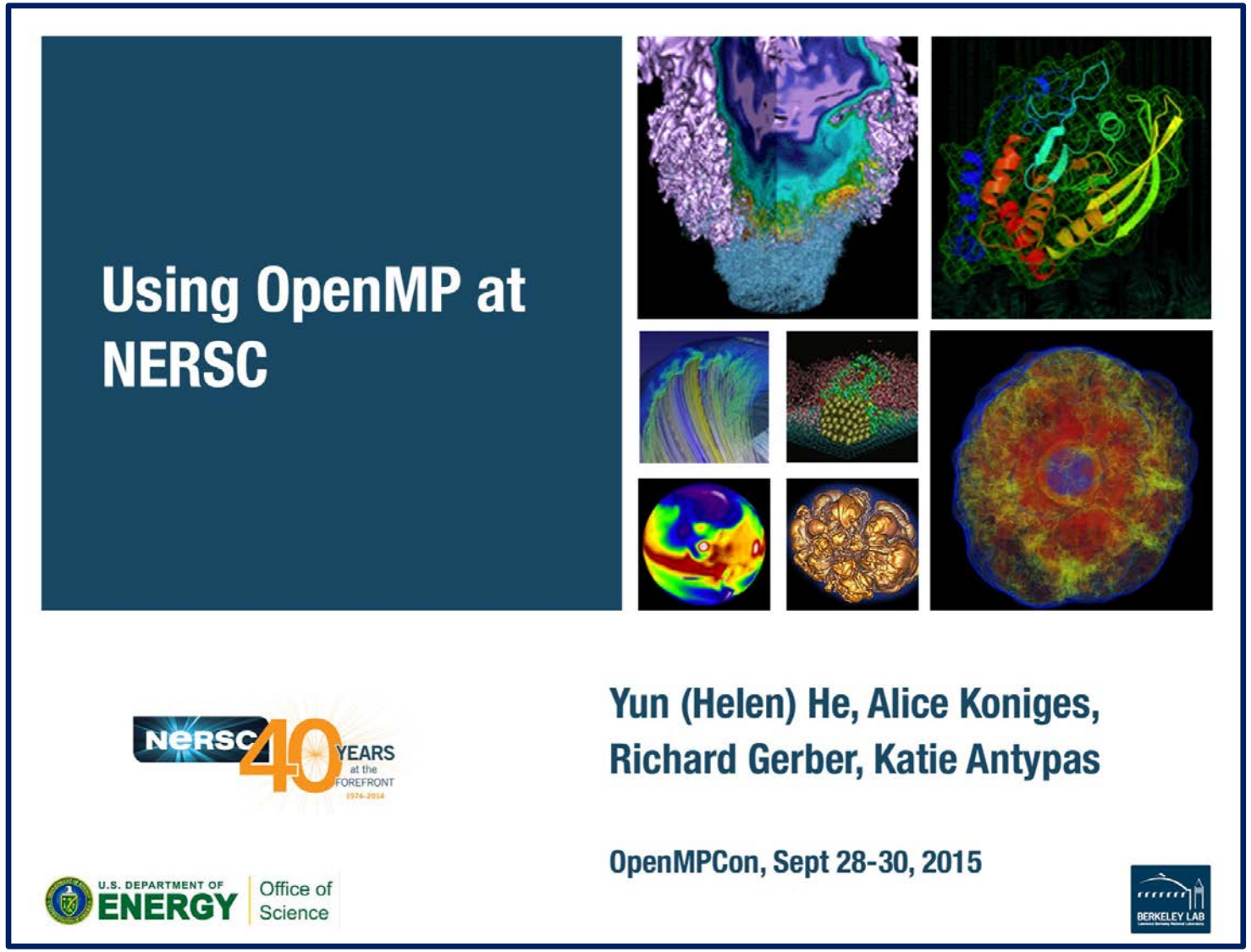
- The fact is, we don't really know how much our programming models are used.
  - Counting downloads or stars on github repositories provides little evidence for how many people are writing code with a programming model.

- We judge success by anecdote and rough guestimates.

Collecting hard data on programming model usage is expensive, time consuming, and requires a great deal of hard work!

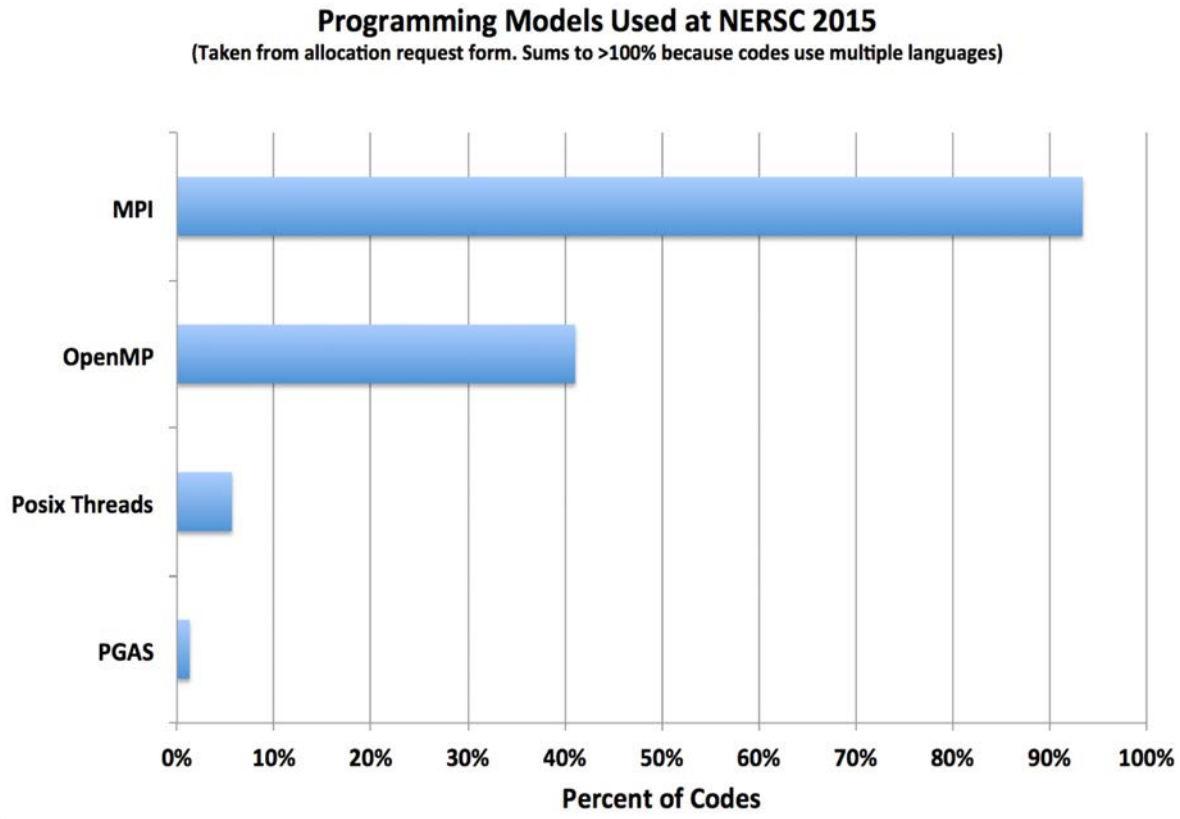
So we don't do it and "trust our gut".

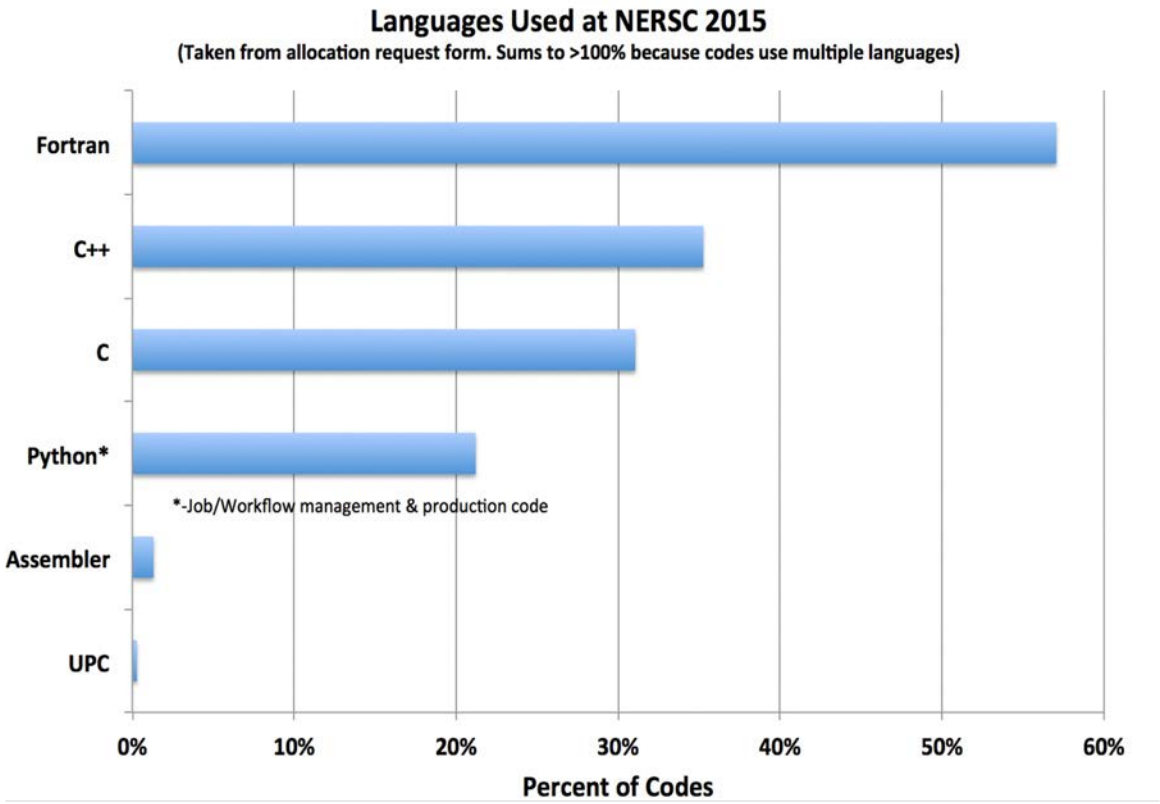
One of the only public sources of hard data we are aware of



Source: https://www.nersc.gov/assets/Uploads/HelenHe-OpenMPCon-2015.pdf

Programming models used at NERSC (2015)





At the time of this study, the production NERSC machines were large systems that emphasized the CPU:

- Edison. Cray SC30, 5,576 nodes with 2 12 core Xeon processors per node (2 NUMA domains).
- Hopper. Cray XE6, 6,384 nodes with 2 12 core Xeon processors per node (4 NUMA domains).

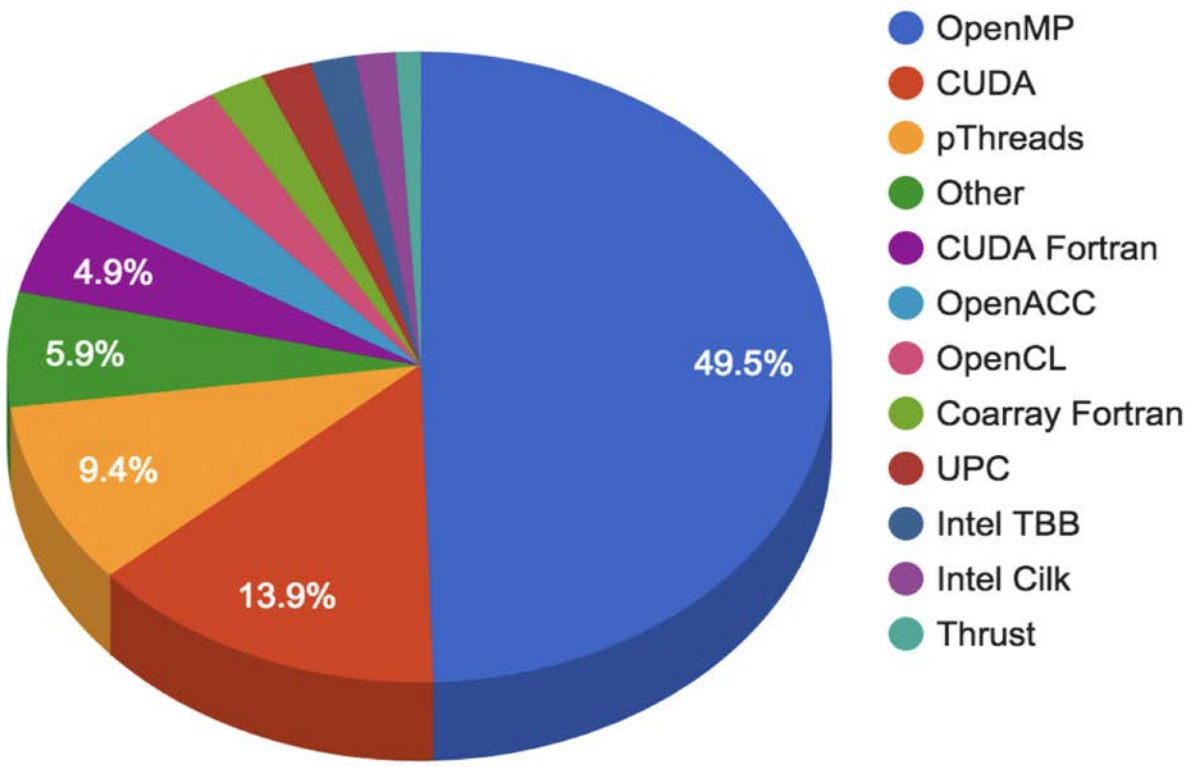
The study was based on 6900 users, over 850 projects working with over 600 codes.

Source: https://www.nersc.gov/assets/Uploads/HelenHe-OpenMPCon-2015.pdf

Programming models used at NERSC (2015)

What's the X in MPI+X?

OpenMP is about 50%, out of all choices of X



Source: https://www.nersc.gov/assets/Uploads/HelenHe-OpenMPCon-2015.pdf

Where can we find more generally applicable data that covers a wider range of systems/applications?

Enter Stage Left …

- Gal Oren at Technion and his students at Ben-Gurion University.
- Research on AI used in Machine programming … in particular AI to help generate OpenMP programs.
- Central hypothesis: Its better to have a small Large-Language-Model trained on data that matches the problem domain … Scope is all you need to get more value from your large language model
- To explore this hypothesis, they collected data restricted to HPC code to be used to train HPC specific large language models. They call this data set **HPCorpus**.

- Tal Kadosh, Niranjan Hasabnis, Vy A. Vo, Nadav Schneider, Neva Krien, Abdul Wasay, Nesreen Ahmed, Ted Willke, Guy Tamir, Yuval Pinter, Timothy Mattson, and Gal Oren . "**Scope is all you need: Transforming LLMs for HPC Code**." *arXiv preprint arXiv:2308.09440* (2023). IXPUG'23
- Tal Kadosh, Nadav Schneider, Niranjan Hasabnis, Timothy Mattson, Yuval Pinter, and Gal Oren. "**Advising OpenMP Parallelization via a Graph-Based Approach with Transformers.**" *arXiv preprint arXiv:2305.11999* (2023). IWOMP'23
- Gal Oren. "**Unlocking the Potential of Large Language Models for High Performance Computing Code.**" *AI Assisted Software Development for HPC (*2023). AI4DEV'23
- Nadav Schneider, Tal Kadosh, Niranjan Hasabnis, Timothy Mattson, Yuval Pinter, and Gal Oren. "**MPI-RICAL: Data-Driven MPI Distributed Parallelism Assistance with Transformers.**" *AI Assisted Software Development for HPC* (2023). AI4DEV'23
- Re'em Harel, Yuval Pinter, and Gal Oren. "**Learning to parallelize in a shared-memory environment with transformers.**" *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming* (2023). PPoPP'23

HPCorpus data collection

Scan all C, C++ and Fortran codes from github with “last updated” dates between 2012 and mid 2023

	Repos	Size(GB)	Files (#)	Functions (#)
C	144,522	46.23	4,552,736	87,817,591
C++	150,481	26.16	4,735,196	68,233,984
Fortran	3,683	0.68	138,552	359,272

```
WITH selected_repos as (
SELECT f.id, f.repo_name as repo_name, f.ref as
  ref, f.path as path
FROM `bigquery-public-data.github_repos .files` as
  f
JOIN `bigquery-public-data.github_repos .licenses`
  as l on l.repo_name = f.repo_name
),
deduped_files as (
SELECT f.id, MIN(f.repo_name) as repo_name,
  MIN(f.ref) as ref, MIN(f.path) as path
FROM selected_repos as f
GROUP BY f.id
)
SELECT
f.repo_name, f.ref, f.path, c.copies, c.content,
FROM deduped_files as f
JOIN `bigquery-public-data.github_repos .contents`
  as c on f.id = c.id
WHERE
 NOT c.binary
 AND (f.path like '%.c' OR f.path like '%.cpp' OR
  f.path like '%.f' OR f.path like '%.f90' OR
  f.path like '%.f95')
```

Query used with Google BigQuery to collect HPCorpus data

HPCorpus data collection

Scan all C, C++ and Fortran codes from github with “last updated” dates between 201

The BIG idea …

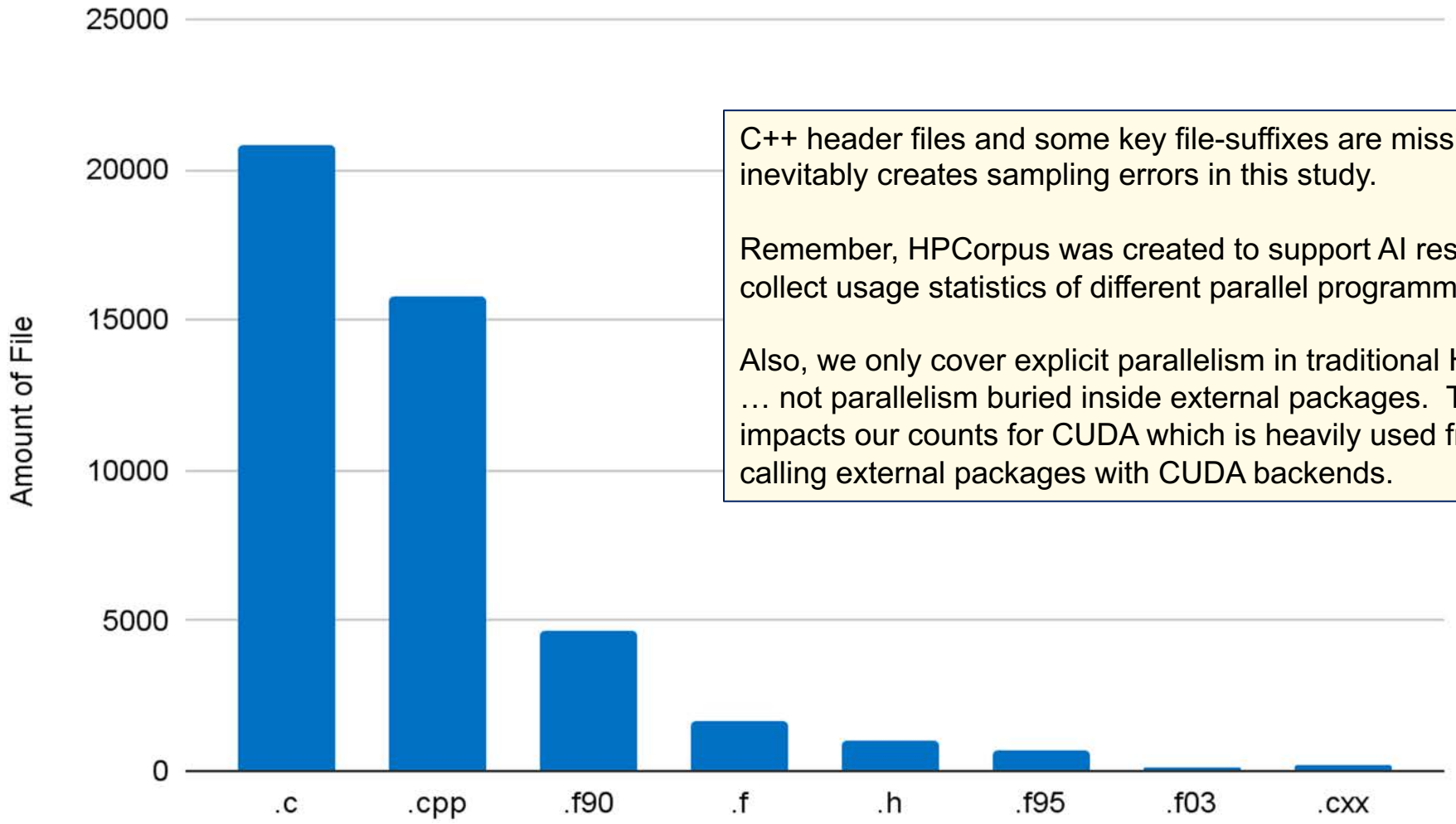
HPCorpus was collected to train AI models for machine programing, but it can also be used to generate quantitative data on Programming model usage in HPC

	Repos			
C	144,522			
C++	150,481	26.16	4,735,196	68,233,984
Fortran	3,683	0.68	138,552	359,272

```
1  WITH selected_repos as (
2  SELECT f.id, f.repo_name as repo_name, f.ref as
   ↪  ref, f.path as path
3  FROM `bigquery-public-data.github_repos .files` as

                    data.github_repos .licenses`
                    e = f.repo_name


                    o_name) as repo_name,
                     MIN(f.path) as path
                     f



                    path, c.copies, c.content,
                    f
14 JOIN `bigquery-public-data.github_repos .contents`
   ↪  as c on f.id = c.id
15 WHERE
16  NOT c.binary
17  AND (f.path like '%.c' OR f.path like '%.cpp' OR
    ↪  f.path like '%.f' OR f.path like '%.f90' OR
    ↪  f.path like '%.f95')
```

Query used with Google BigQuery to collect HPCorpus data

Language/file-type breakdown for files in HPCorpus



C++ header files and some key file-suffixes are missing … this inevitably creates sampling errors in this study.

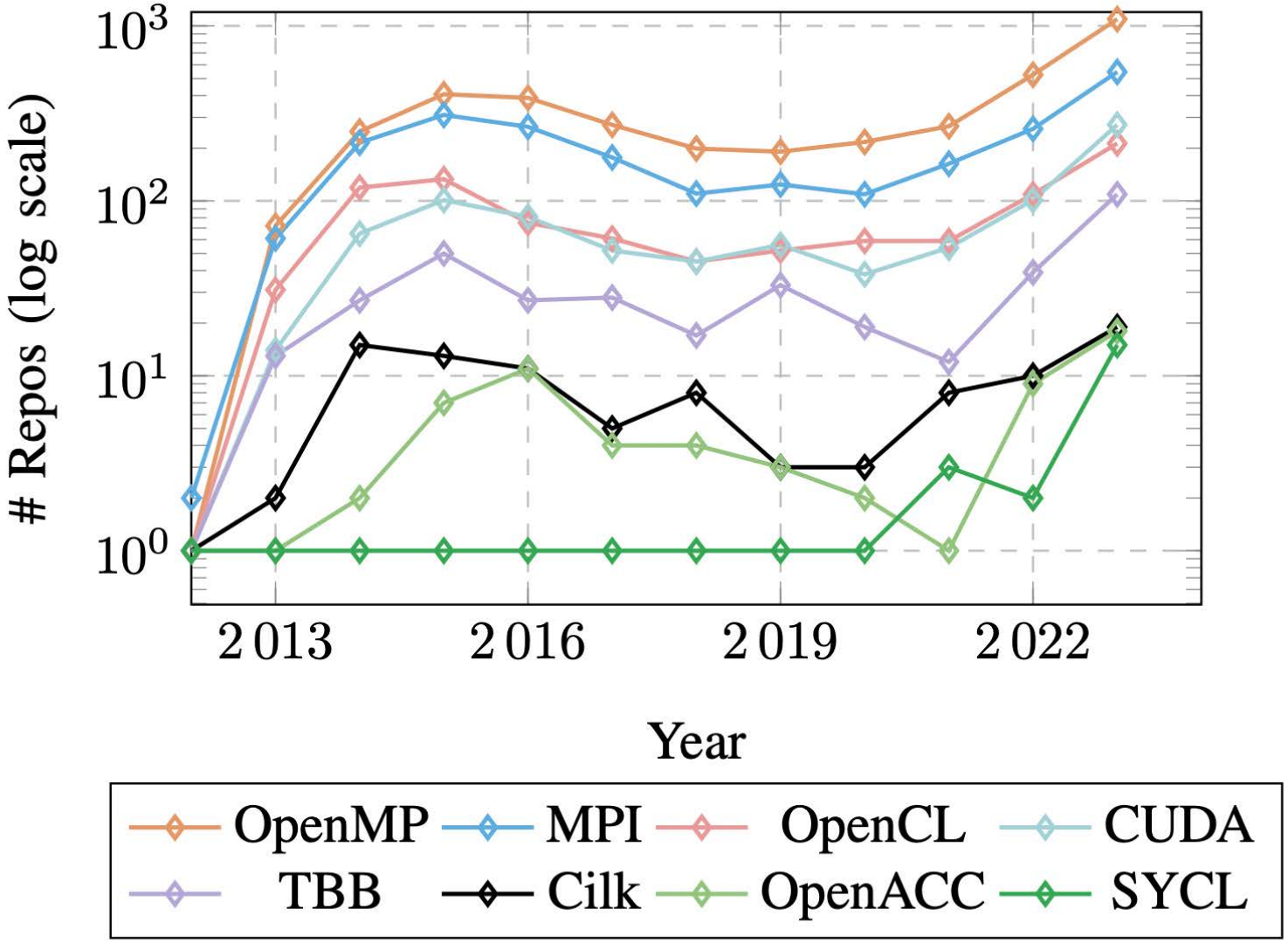
Remember, HPCorpus was created to support AI research, not to collect usage statistics of different parallel programming models.

Also, we only cover explicit parallelism in traditional HPC languages … not parallelism buried inside external packages. This particular impacts our counts for CUDA which is heavily used from python calling external packages with CUDA backends.

Note: since we did not collect files with .cu or .cuf suffices, we undercounted CUDA usage in HPCorpus.

So ... how popular is OpenMP?

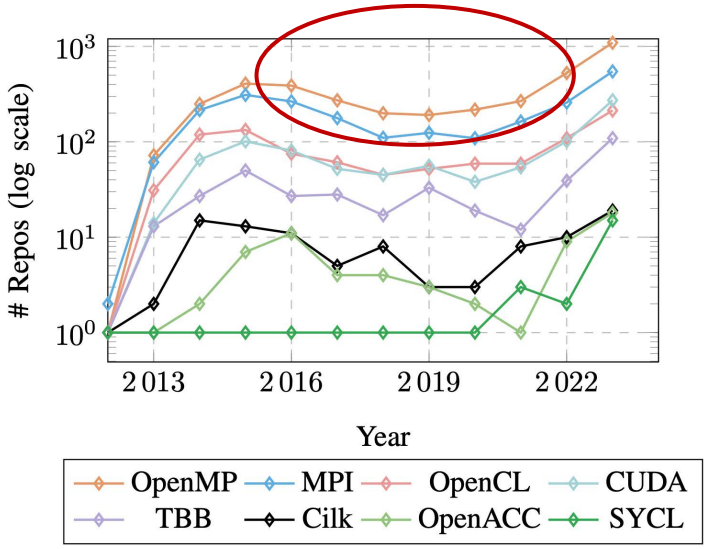
Programming models usage 2013 to 2023



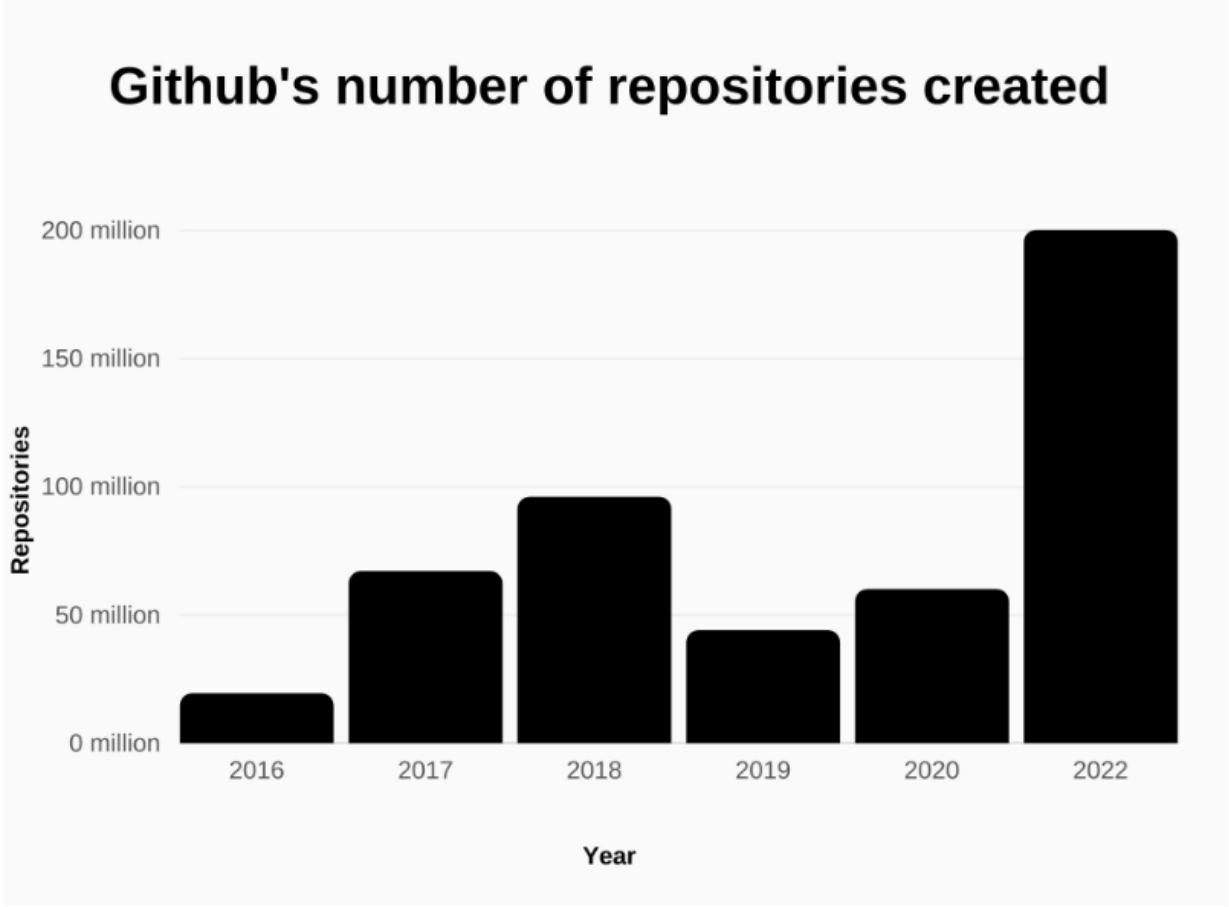
Note: since we did not collect files with .cu or .cuf suffices, we undercounted CUDA usage in HPCorpus.

The years in this plot indicate the year the repository was last updated

Why the dip from 2016 to 2019?

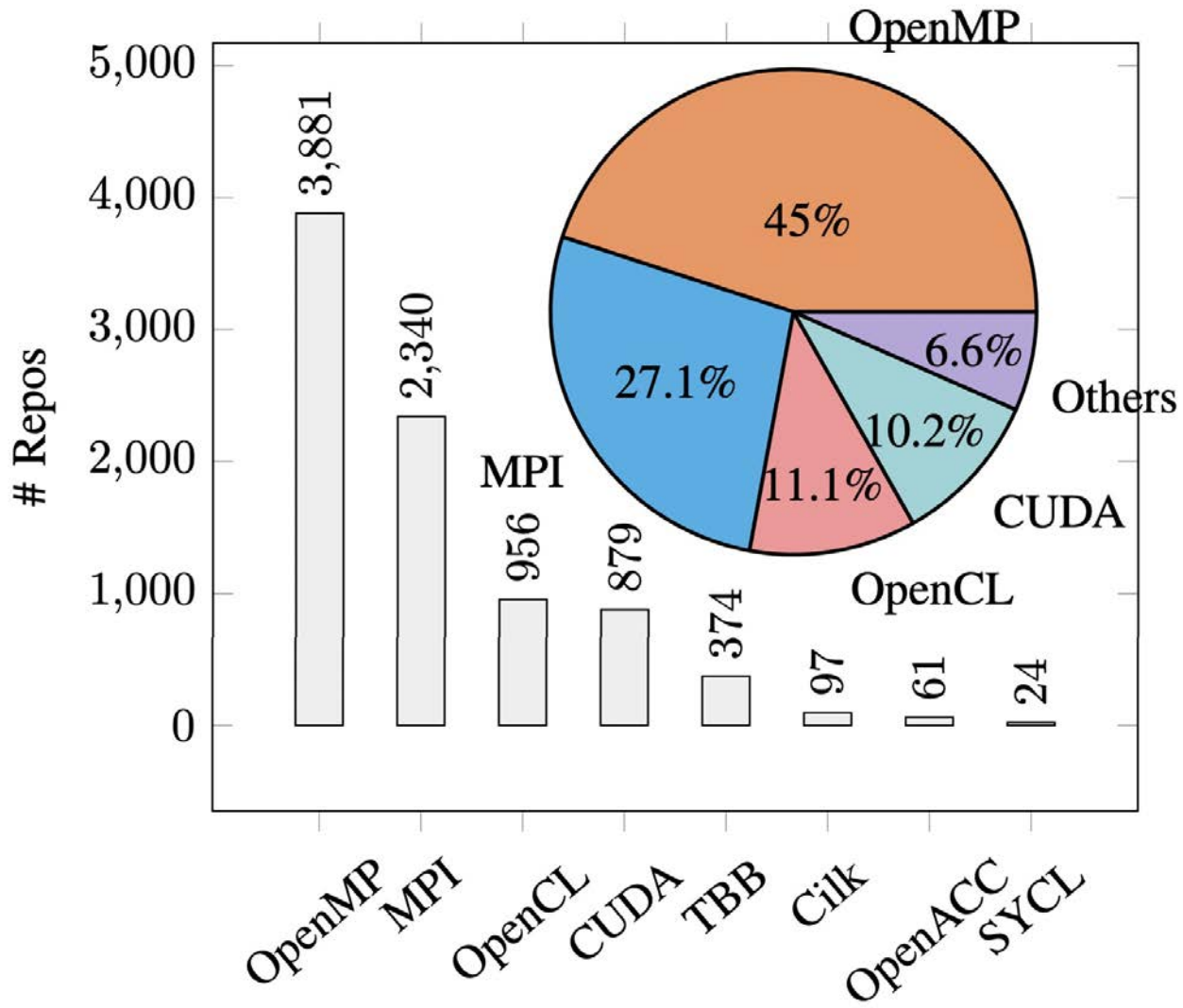


We don't know why … but it is similar to patterns in gitHub repository creation between 2018 and 2022



https://www.usesignhouse.com/blog/github-stats. downloaded Oct 20, 2023

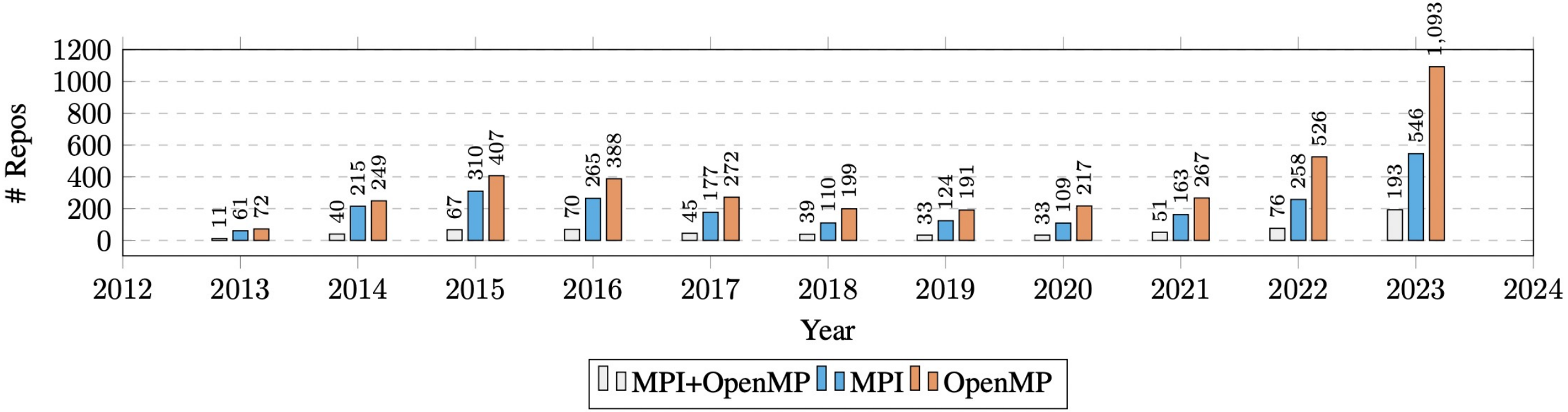
Programming models usage 2013 to 2023



Note: since we did not collect files with .cu or .cuf suffices, we undercounted CUDA usage in HPCorpus.

Aggregate numbers over all repositories from 2013 to 2023

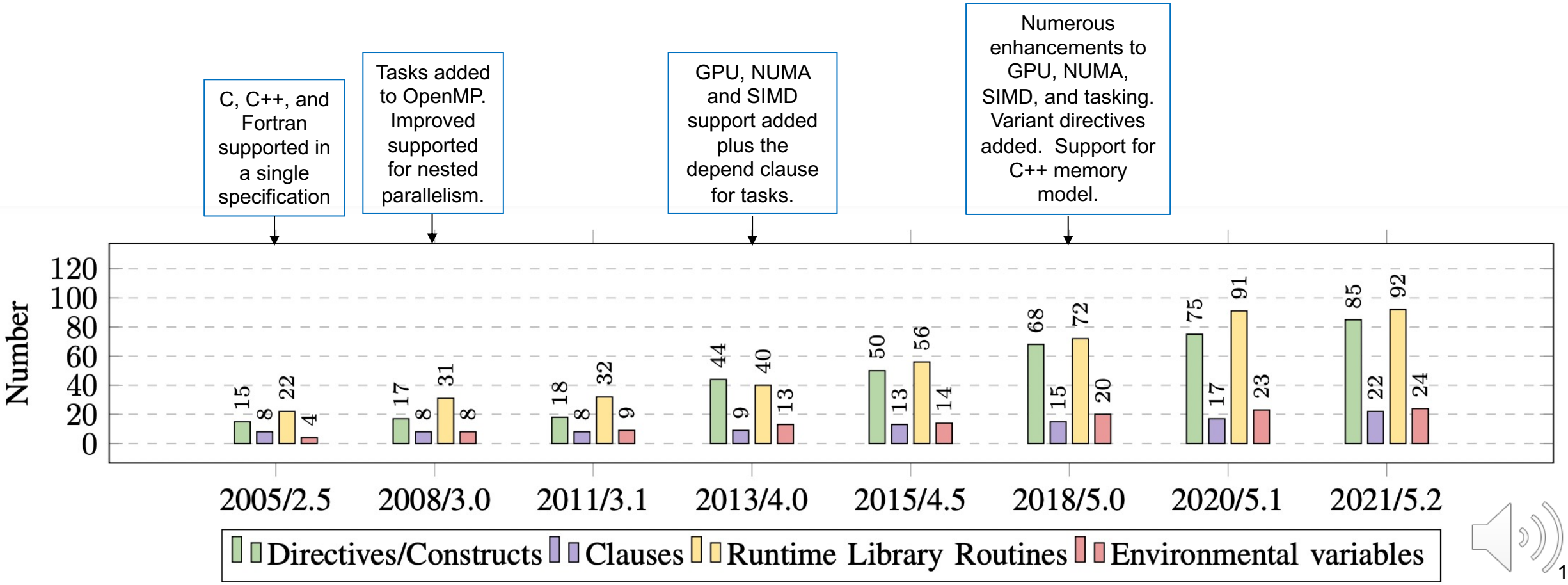
Comparing OpenMP, MPI and MPI+OpenMP



The years in this plot indicate the year the repository was last updated

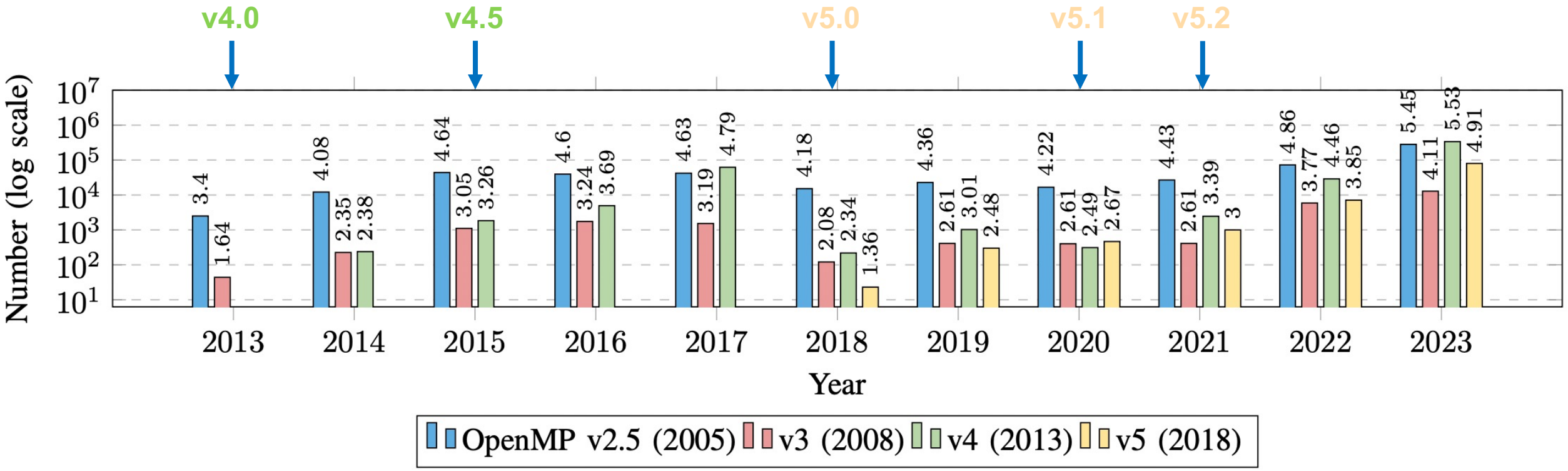
OpenMP Specifications

- As the range of algorithms addressed by OpenMP increases and new hardware platforms are supported, the specifications grow.
- In the following, we list the total number of directives, clauses, runtime library routines and environment variables for each version of the specification. We also highlight major changes with certain specifications.



Adoption of new OpenMP Specifications

- We can track constructs added with each version of the OpenMP specification to see how quickly the new specs are being adopted.

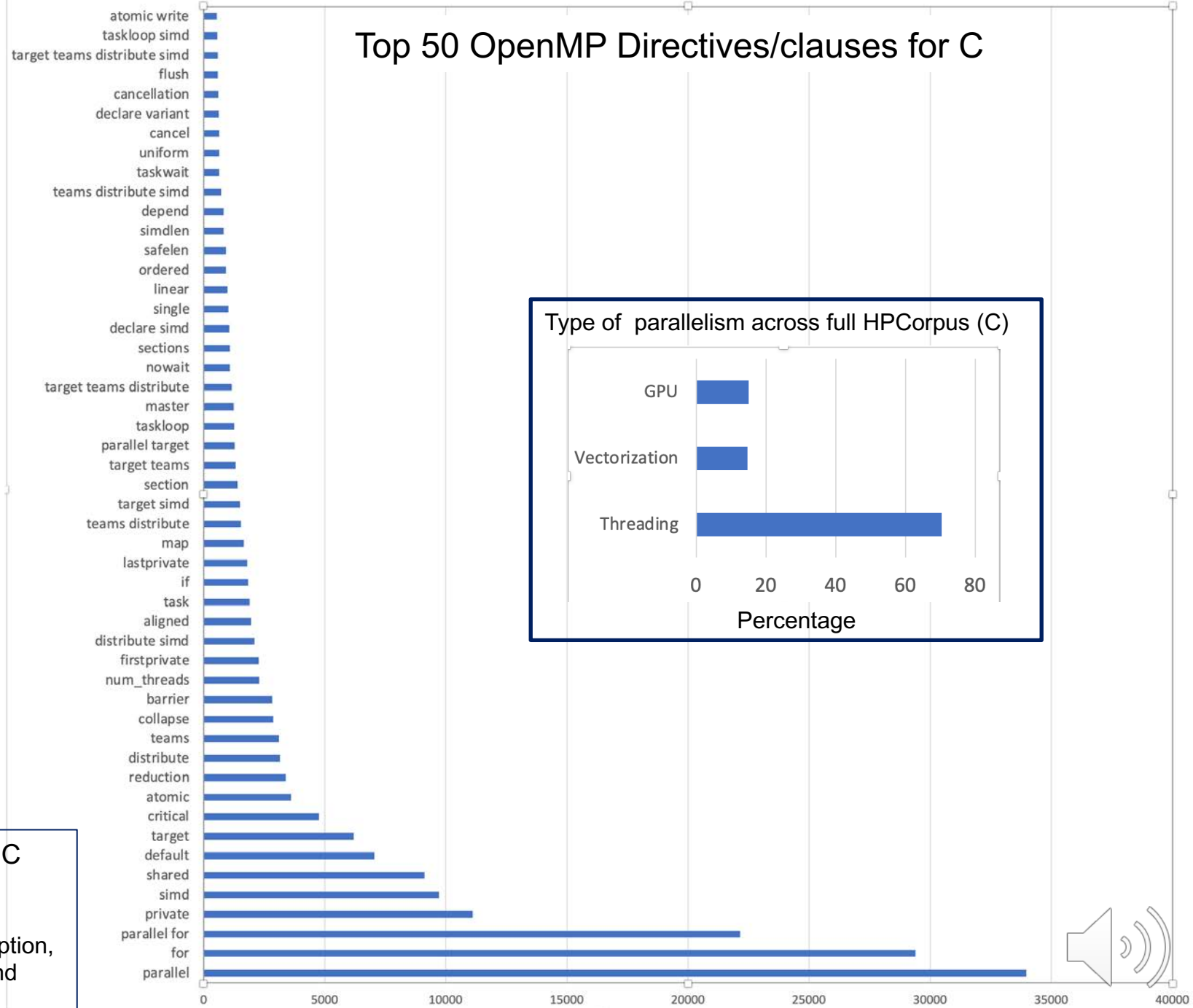


The years in this plot indicate the year the repository was last updated

What are people actually using from OpenMP

With the HPCorpus* dataset, we finally have hard-data to analyze what “should” be in the common core.

This data was constructed by summing up counts for different directives and clauses across time from 2013 to the middle of 2023.

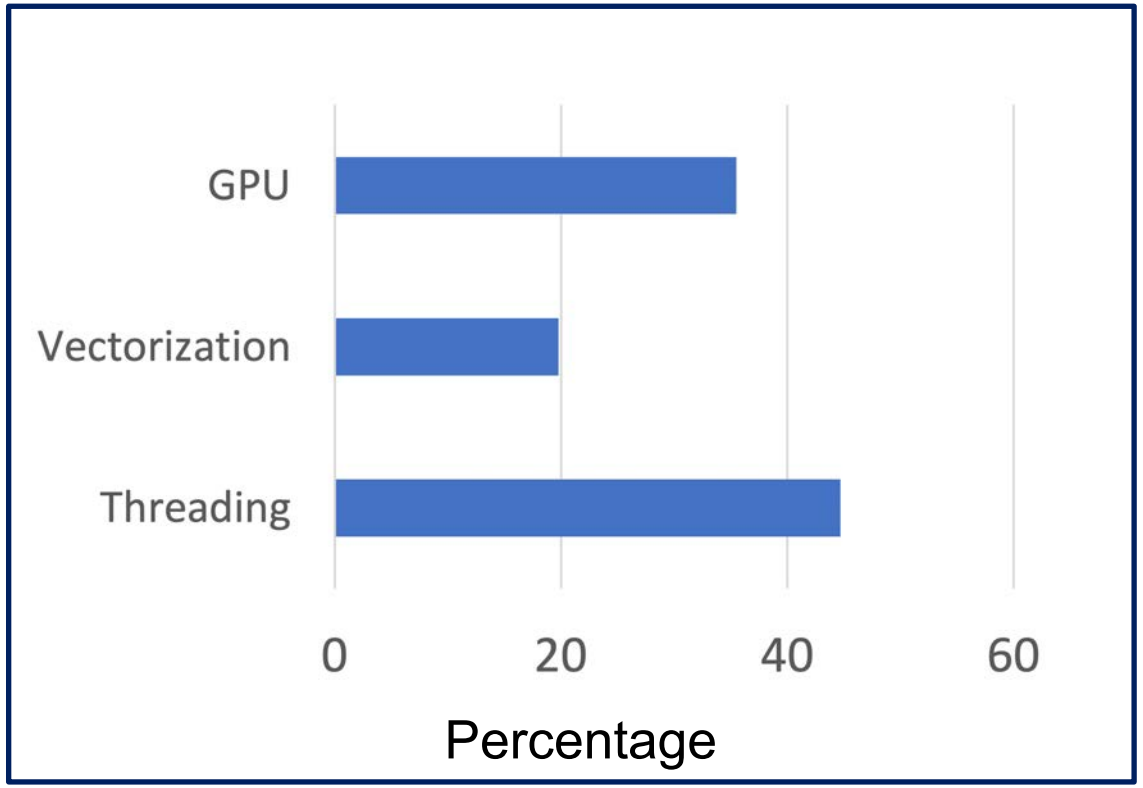


HPCorpus … a data set created by scraping ”all” HPC codes from github written in C, C++ and Fortran.

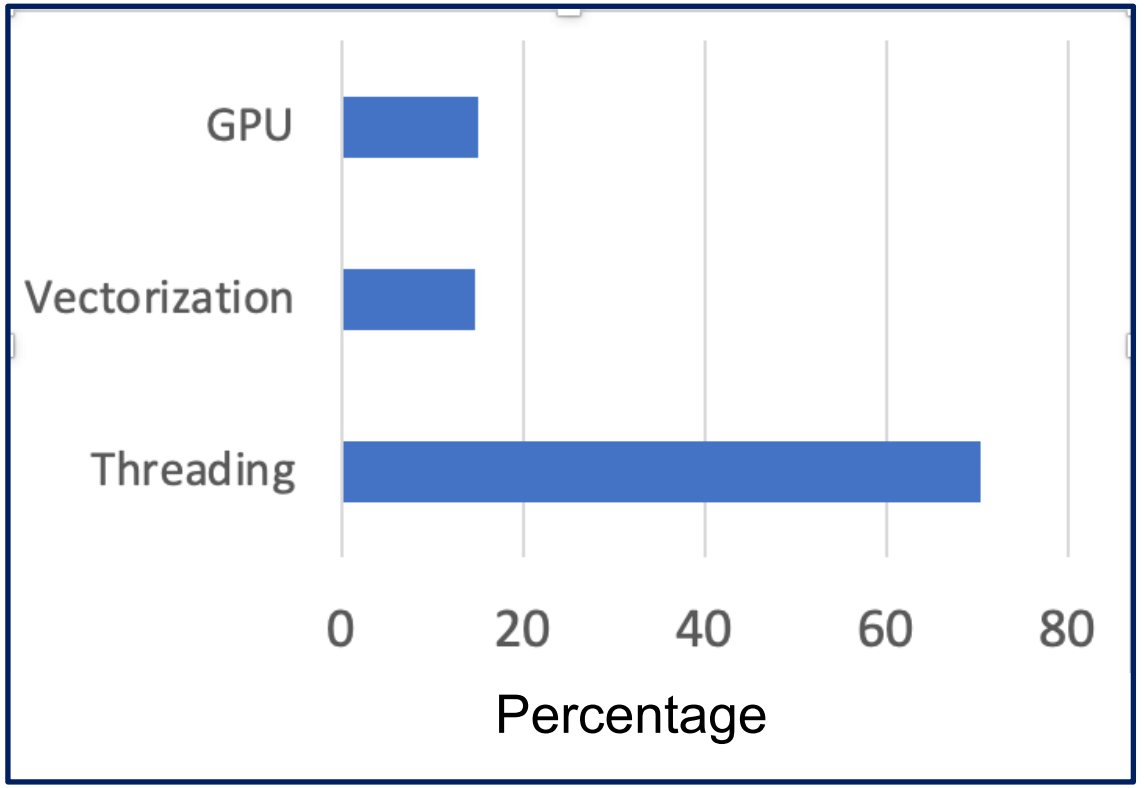
Quantifying OpenMP: Statistical Insights into Usage and Adoption, Tal Kadosh, Niranjan Hasabnis, Tim Mattson, Yuval Pinter, and Gal Oren, IEEE HPEC 2023

C++ vs C OpenMP programs in HPCorpus

Type of parallelism across full HPCorpus (C++)

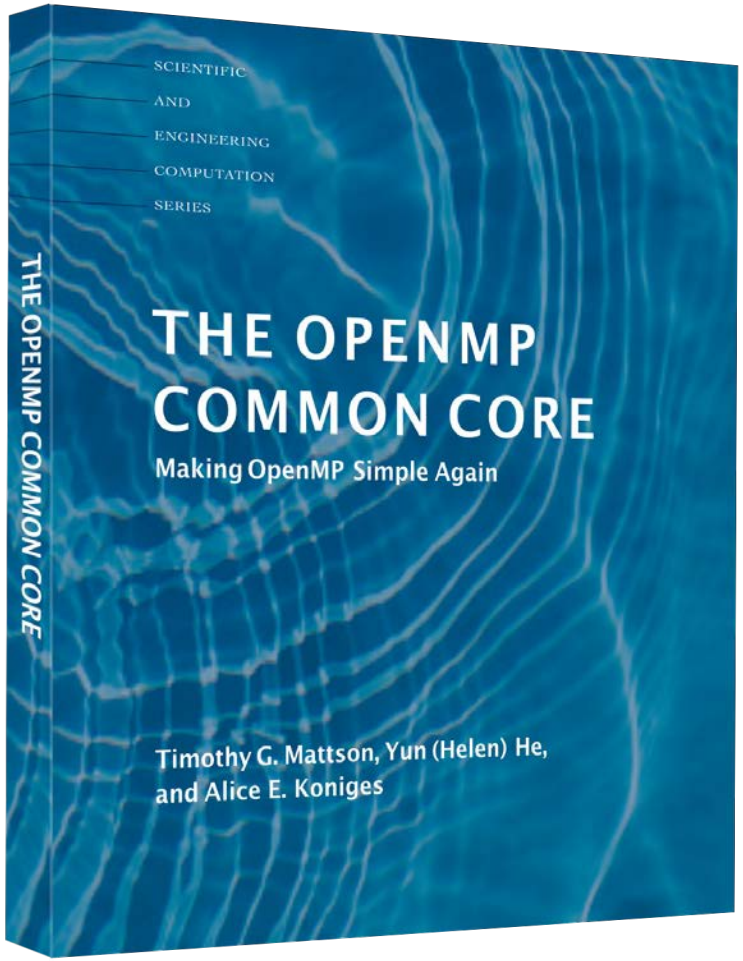


Type of parallelism across full HPCorpus (C)



C++ programmers are MUCH more aggressive about the newer types of parallelism in OpenMP

The OpenMP Common Core



For many years now, we've been teaching the subset of OpenMP that is most commonly used. We call this the

OpenMP Common Core

We even wrote a book about it.

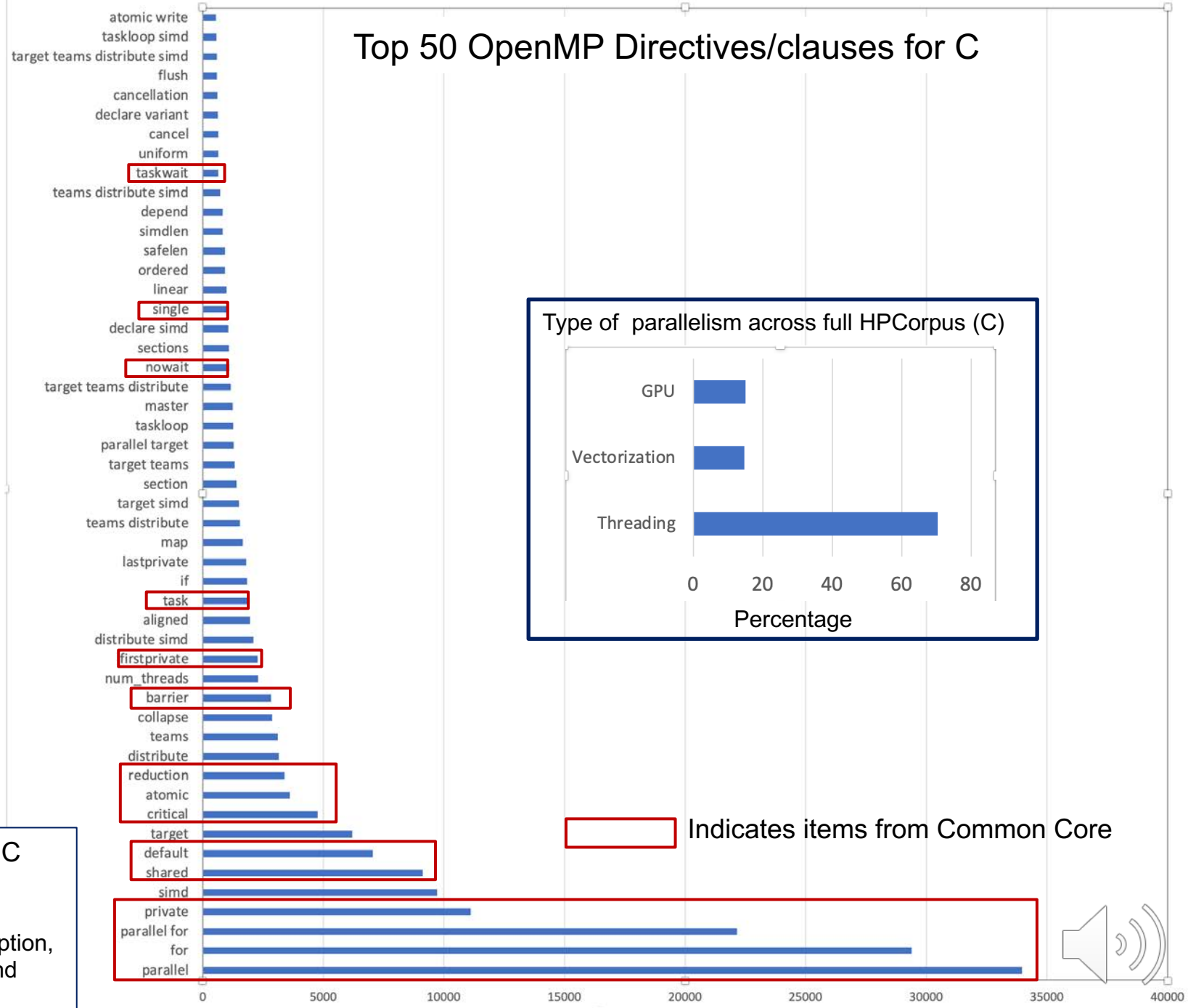
The list of items in the common core were determined by experience/anecdote … we didn't have hard data to drive the analysis.

The OpenMP Common Core
#pragma omp parallel
void omp_set_thread_num() int omp_get_thread_num() int omp_get_num_threads()
double omp_get_wtime()
setenv OMP_NUM_THREADS N
#pragma omp barrier #pragma omp critical
#pragma omp for #pragma omp parallel for
reduction(op:list)
schedule (static [,chunk]) schedule(dynamic [,chunk])
shared(list), private(list), firstprivate(list)
default(none)
nowait
#pragma omp single
#pragma omp task #pragma omp taskwait

What are people actually using from OpenMP

With the HPCorpus* dataset, we finally have hard-data to analyze what “should” be in the common core.

This data was constructed by summing up counts for different directives and clauses across time from 2013 to the middle of 2023.



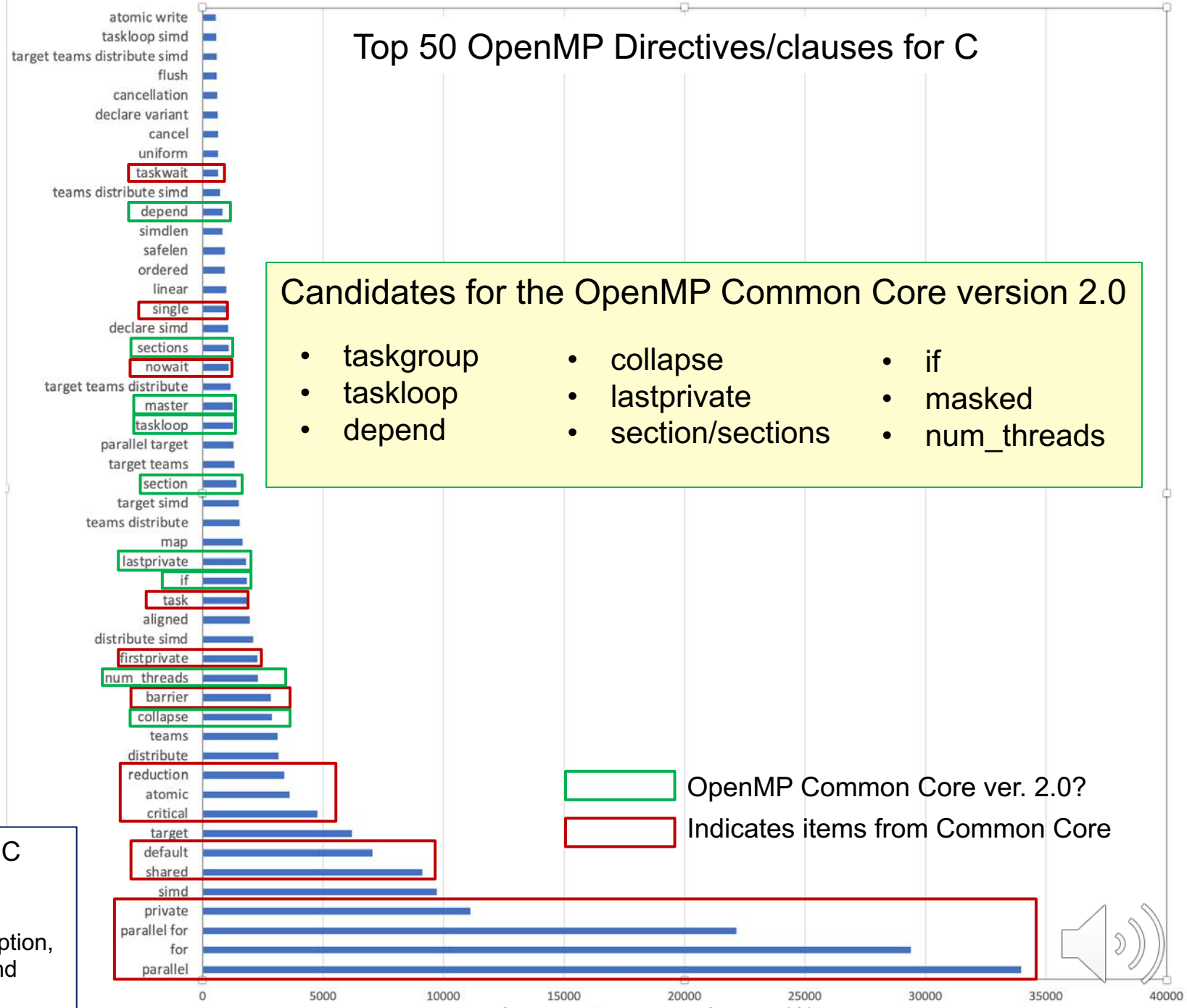
HPCorpus … a data set created by scraping ”all” HPC codes from github written in C, C++ and Fortran.

Quantifying OpenMP: Statistical Insights into Usage and Adoption, Tal Kadosh, Niranjan Hasabnis, Tim Mattson, Yuval Pinter, and Gal Oren, IEEE HPEC 2023

What are people actually using from OpenMP

With the HPCorpus* dataset, we finally have hard-data to analyze what “should” be in the common core.

This data was constructed by summing up counts for different directives and clauses across time from 2013 to the middle of 2023.



HPCorpus … a data set created by scraping ”all” HPC codes from github written in C, C++ and Fortran.

Quantifying OpenMP: Statistical Insights into Usage and Adoption, Tal Kadosh, Niranjan Hasabnis, Tim Mattson, Yuval Pinter, and Gal Oren, IEEE HPEC 2023

Conclusion

- Kayaking is a wonderful way to experience this beautiful world we live in.
- OpenMP is the number one parallel programming model in use today.
- The C++ community is much more aggressive with using the newer features (SIMD and GPU) in OpenMP.
- The HPCorpus data set is amazing and publicly available:

  https://github.com/Scientific-Computing-Lab-NRCN/HPCorpus

  - Credit for HPCorpus goes to Gal Oren* and his team (Tal Kadosh and Yuval Pinter).
- Future work:
  - Build machine programming systems using AI models trained with HPCorpus.
  - Apply the same approach used here for OpenMP to MPI and TBB

* Scientific Computing Lab. Head: Gal Oren
https://github.com/Scientific-Computing-Lab-NRCN