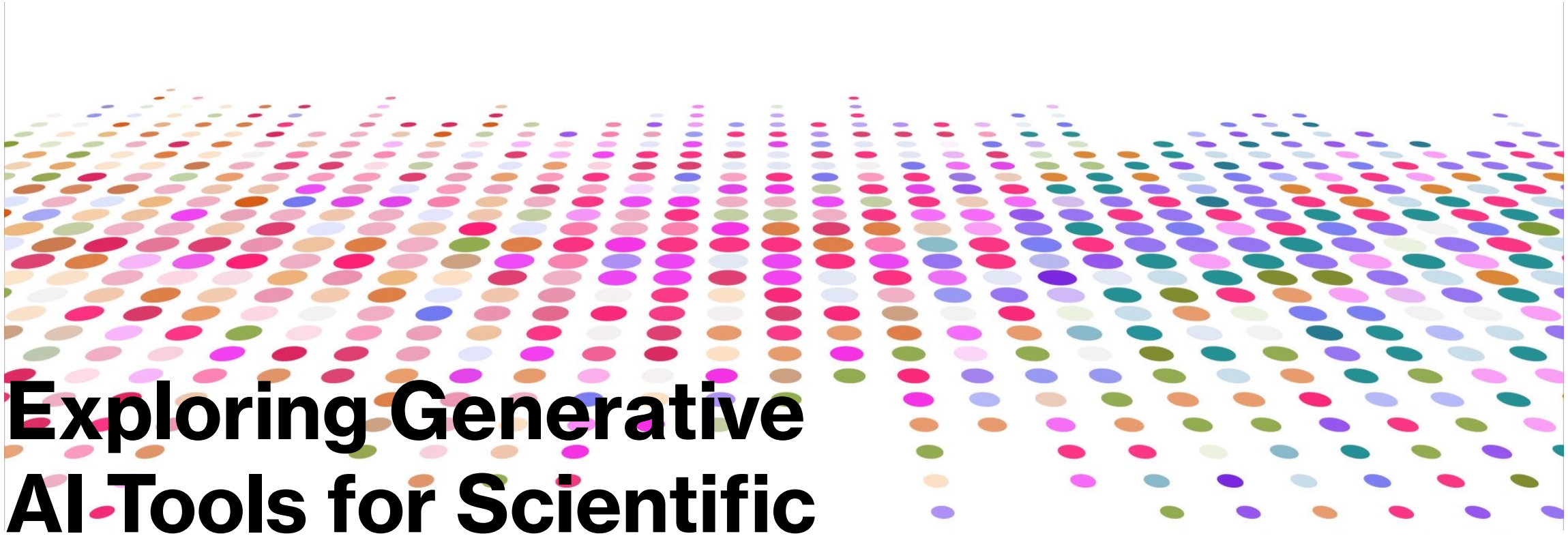


Exploring Generative AI Tools for Scientific Software Development

First ideas and experiences

Michael A. Heroux
Siva Rajamanickam

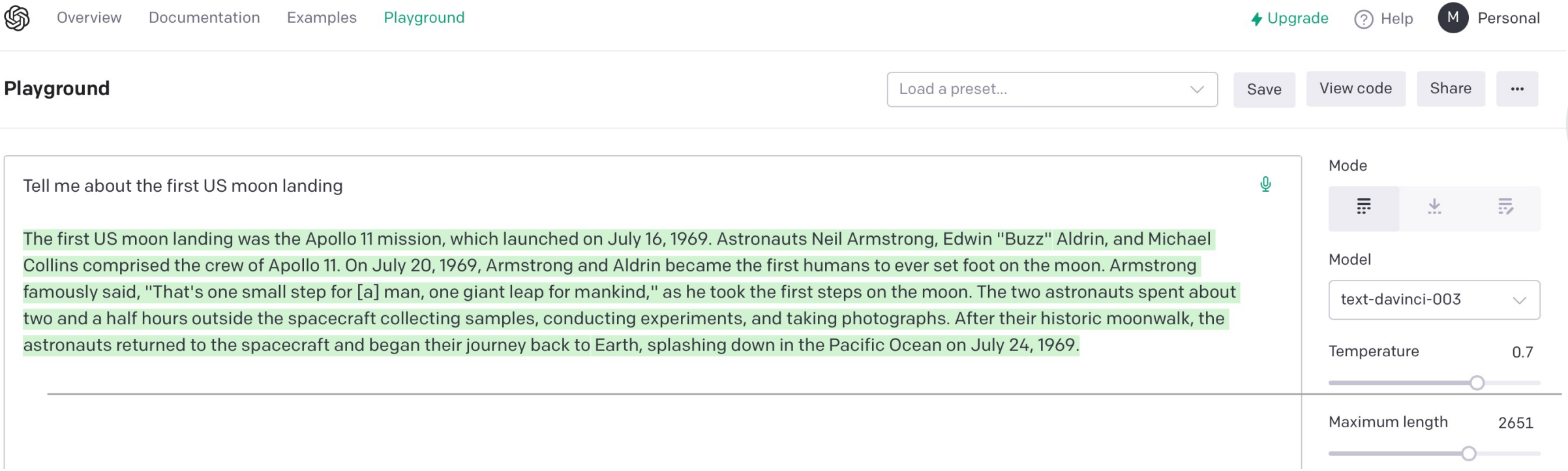
ECP Community BOF Days 2023

BOF Purpose

- Provide some brief examples of use
- Speculate on possible opportunities/challenges
- Real purpose:
  - Connect with people interested in generative AI
  - Start a conversation going forward
- Interested in further discussion?
  - See Google Sheet sign up in Zoom chat
  - And here: https://docs.google.com/spreadsheets/d/17NdjprJr0-lVmoJjitIeEmgITfJRIduBoWO8jNX2b9M/edit?usp=sharing

What are Generative AI Tools?

- Chat:
  - Produce content by
    - Accepting an input prompt
    - Producing human-like output in response to the prompt
  - Example:
    - Prompt: *Tell me about the first US moon landing*
    - https://beta.openai.com – Playground
- Inline suggested content:
  - Predict what you need next as you type
  - Example: GitHub Copilot auto-complete

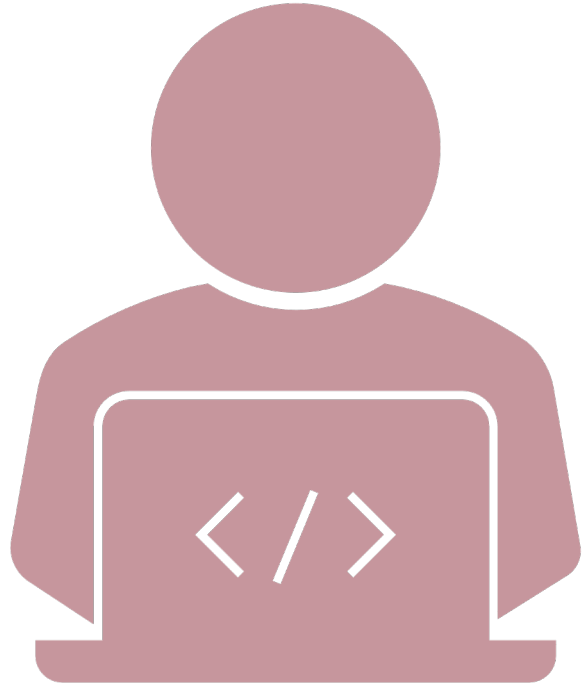


Tools I have used

- OpenAI: Company producing most-used tools
  - GitHub Copilot:
    - Have been tracking since Dec 2021
    - Based on Codex: special OpenAI model for programming
    - Available as a plugin for VSCode (how I use it)
  - OpenAI Playground:
    - Like ChatGPT with more flexible, complicated API (to GPT3)
    - Less known, more available
  - ChatGPT:
    - Latest API (to GPT3.5)
    - Raised broad awareness.
- Not tried: Jasper (also based on GPT3), Code Whisperer (Amazon)
- Many new tools emerging and expected in the coming months

Some ways to think about these tools

- College roommate English major who lives to help you with your writing assignments
- Programmer to produce code, scripts, templates as starting point for you to consider
- Administrative assistant to help with detailed content formatting tasks
- Elaborate auto-complete feature
- An on-demand intelligence



English major roommate

- Authoring
  - Brainstorm topics
  - Generate outline
  - Recursively provide more detail
- Assessment
  - Request AI review of content
- Improvement
  - Steel-manning other perspectives
  - Refine content – understandable by a novice, TL;DR, etc.

What does it mean to be the author?

- Student essay due by midnight:
  - Written with copy-and-paste and “lite” editing after dinner
  - Is the student the author?
- Definition of author here:
  - Ability to explain, defend, and elaborate on written content?
  - True novelty seems unlikely
- How to assess knowledge:
  - Assessing quality of writing OK but not content knowledge
  - Assessment via dialogue seems essential

Basic rules for using sophisticated (including AI) tools in content development – 2 versions

1. If used to assist creating/transforming **your own content**:
   a) Make sure you truly understand/own the content
   b) Beware of plagiarism
2. If used to incorporate **others' content**:
   a) Assure rights to use
   b) Give credit
3. Do not list tools as authors

1. Report the use of these tools if used to create content
2. Assure that all authors take responsibility for content
3. Do not list tools as authors

Skills impact

- Low-level skills replaced by tools
  - Programming:
    - Syntax, formatting
    - Programming standards
    - Using APIs
- What to produce and how it’s designed are relatively more important

Trends

- Soft skills become relatively more important
  - Understanding the individual, teams, communities become more important that technical skills
- Domain knowledge relatively more important
  - What to produce – Requirements
- Design become more important & easier
  - Rapid prototyping easier
- AI tools won’t replace people
  - People who use AI tools will replace those who don’t

https://beta.openai.com/playground
Bootstrapping a scripting tool – Step 1: Ask for advice

What is a good way to extract metadata from a GitHub repo?

The best way to extract metadata from a GitHub repo is to use the GitHub API. It provides a range of endpoints that allow you to access information about the repo, such as the number of commits, pull requests, releases, and other metadata. You can also use the GitHub GraphQL API to query more specific information.

Write a script to extract basic activity information from the Trilinos GitHub repo

```
#!/usr/bin/env python3

import requests

# Enter your GitHub personal access token here
token = '<YOUR_TOKEN_HERE>'

# The GitHub repository to query
repo = 'trilinos/Trilinos'

# Get the list of commits
url = f'https://api.github.com/repos/{repo}/commits'
headers = {'Authorization': f'token {token}'}
response = requests.get(url, headers=headers).json()

# Print the commit count
print(f'Commit count: {len(response)}')

# Get the list of pull requests
url = f'https://api.github.com/repos/{repo}/pulls'
headers = {'Authorization': f'token {token}'}
response = requests.get(url, headers=headers).json()

# Print the pull request count
print(f'Pull request count: {len(response)}')

# Get the list of releases
url = f'https://api.github.com/repos/{repo}/releases'
headers = {'Authorization': f'token {token}'}
response = requests.get(url, headers=headers).json()

# Print the release count
print(f'Release count: {len(response)}')
```

https://beta.openai.com/playground

Step 2: Generate script

https://beta.openai.com/playground

Step 3: Learn about token

How do I determine my github personal access token?

To generate a GitHub personal access token, you must first log in to your GitHub account. Once logged in, go to your Settings page, select “Developer settings” from the left sidebar, and then “Personal access tokens”. On the next page, click “Generate new token”. You will then be prompted to give your token a name and select the scopes for which the token will be granted access. Once you have set your token’s name and scopes, click “Generate token”. Your token will then be displayed on the next page.

https://beta.openai.com/playground

Step 4: Experiment

Write a github API python script to list the contributors and number of commits to a github repo with user provided name

```
import requests
import json


# Get the user input
user_input = input("Please enter the GitHub repo name: ")


# API call to get the list of contributors
url = "https://api.github.com/repos/{}/contributors".format(user_input)
r = requests.get(url)


# Parse the response
data = json.loads(r.text)


# Print the result
print("List of Contributors and Number of Commits to {}".format(user_input))
for contributor in data:
    print("{}: {}".format(contributor['login'], contributor['contributions']))
```

VSCode+Copilot
HPCG Benchmark WAXPBY Kernel

ComputeWAXPBY_ref.cpp 2

src > ComputeWAXPBY_ref.cpp > ...

```
41  */
42  int ComputeWAXPBY_ref(const local_int_t n, const double alpha, const Vector & x,
43      const double beta, const Vector & y, Vector & w) {
44
45    assert(x.localLength>=n); // Test vector lengths
46    assert(y.localLength>=n);
47
48    const double * const xv = x.values;
49    const double * const yv = y.values;
50    double * const wv = w.values;
51
52    if (alpha==1.0) {
53  #ifndef HPCG_NO_OPENMP
54      #pragma omp parallel for
55  #endif
56      for (local_int_t i=0; i<n; i++) wv[i] = xv[i] + beta * yv[i];
57    } else if (beta==1.0) {
58  #ifndef HPCG_NO_OPENMP
59      #pragma omp parallel for
60  #endif
61      for (local_int_t i=0; i<n; i++) wv[i] = alpha * xv[i] + yv[i];
62    } else  {
63  #ifndef HPCG_NO_OPENMP
64      #pragma omp parallel for
65  #endif
66      for (local_int_t i=0; i<n; i++) wv[i] = alpha * xv[i] + beta * yv[i];
67    }
68
69    return 0;
70  }
71
```

VSCode with GitHub Copilot Plugin

- Copilot aware of complete HPCG code base
- Copilot also trained on many specialized APIs:
  - Intel Vector ISA
  - Kokkos Performance Portability API
  - Apple, HIP, etc…
- Copilot generates equivalent versions of C++ code
- Comment starts the generation process

VSCode+Copilot
HPCG Benchmark WAXPBY Kernel

Intel_ComputeWAXPBY.cpp 2, U

src > Intel_ComputeWAXPBY.cpp

```
1  // generate computeWAXPBY_ref function using Intel vector intrinsics:
2  int ComputeWAXPBY_ref(const local_int_t n, const double alpha, const Vector & x,
       const double beta, const Vector & y, Vector & w) {

    assert(x.localLength>=n); // Test vector lengths
    assert(y.localLength>=n);

    const double * const xv = x.values;
    const double * const yv = y.values;
    double * const wv = w.values;

    if (alpha==1.0) {
      if (beta==1.0) {
        // w = x + y
        for (local_int_t i=0; i<n; i+=4) {
          __builtin_prefetch(&xv[i+16], 0, 0);
          __builtin_prefetch(&yv[i+16], 0, 0);
          __builtin_prefetch(&wv[i+16], 1, 1);
          const __m128d vx = _mm_load_pd(&xv[i]);
          const __m128d vy = _mm_load_pd(&yv[i]);
          const __m128d vw = _mm_add_pd(vx, vy);
          _mm_store_pd(&wv[i], vw);
        }
      } else {
        // w = x + beta * y
        const __m128d vb = _mm_set1_pd(beta);
        for (local_int_t i=0; i<n; i+=4) {
          __builtin_prefetch(&xv[i+16], 0, 0);
          __builtin_prefetch(&yv[i+16], 0, 0);
          __builtin_prefetch(&wv[i+16], 1, 1);
          const __m128d vx = _mm_load_pd(&xv[i]);
          const __m128d vy = _mm_load_pd(&yv[i]);
          const __m128d vw = _mm_add_pd(vx, _mm_mul_pd(vb, vy));
          _mm_store_pd(&wv[i], vw);
        }
      }
```

VSCode with GitHub Copilot Plugin

- Type comment (green text)
- Wait for Copilot to generate code
- Tab to accept

VSCode+Copilot
HPCG Benchmark WAXPBY Kernel

Intel_ComputeWAXPBY.cpp 9+, U

src > Intel_ComputeWAXPBY.cpp

```cpp
// generate computeWAXPBY_ref function using Intel vector intrinsics:
int ComputeWAXPBY_ref(const local_int_t n, const double alpha, const Vector & x,
    const double beta, const Vector & y, Vector & w) {

  assert(x.localLength>=n); // Test vector lengths
  assert(y.localLength>=n);

  const double * const xv = x.values;
  const double * const yv = y.values;
  double * const wv = w.values;

  if (alpha==1.0) {
    if (beta==1.0) {
      // w = x + y
      for (local_int_t i=0; i<n; i+=4) {
        __builtin_prefetch(&xv[i+16], 0, 0);
        __builtin_prefetch(&yv[i+16], 0, 0);
        __builtin_prefetch(&wv[i+16], 1, 1);
        const __m128d vx = _mm_load_pd(&xv[i]);
        const __m128d vy = _mm_load_pd(&yv[i]);
        const __m128d vw = _mm_add_pd(vx, vy);
        _mm_store_pd(&wv[i], vw);
      }
    } else {
      // w = x + beta * y
      const __m128d vb = _mm_set1_pd(beta);
      for (local_int_t i=0; i<n; i+=4) {
        __builtin_prefetch(&xv[i+16], 0, 0);
        __builtin_prefetch(&yv[i+16], 0, 0);
        __builtin_prefetch(&wv[i+16], 1, 1);
        const __m128d vx = _mm_load_pd(&xv[i]);
        const __m128d vy = _mm_load_pd(&yv[i]);
        const __m128d vw = _mm_add_pd(vx, _mm_mul_pd(vb, vy));
        _mm_store_pd(&wv[i], vw);
      }
    }
```

VSCode+Copilot
HPCG Benchmark WAXPBY Kernel

Kokkos_ComputeWAXPBY.cpp 1, U

src > Kokkos_ComputeWAXPBY.cpp

```
1  // generate computeWAXPBY function using Kokkos::view and Kokkos::parallel_for
2  int ComputeWAXPBY(const local_int_t n, const double alpha, const Vector & x,
       const double beta, const Vector & y, Vector & w) {

     assert(x.localLength>=n); // Test vector lengths
     assert(y.localLength>=n);

     const double * const xv = x.values;
     const double * const yv = y.values;
     double * const wv = w.values;

     if (alpha==1.0) {
       if (beta==1.0) {
         // w = x + y
         Kokkos::parallel_for(n, KOKKOS_LAMBDA (const local_int_t i) {
           wv[i] = xv[i] + yv[i];
         });
       } else {
         // w = x + beta * y
         Kokkos::parallel_for(n, KOKKOS_LAMBDA (const local_int_t i) {
           wv[i] = xv[i] + beta * yv[i];
         });
       }
     } else if (beta==1.0) {
       // w = alpha * x + y
       Kokkos::parallel_for(n, KOKKOS_LAMBDA (const local_int_t i) {
         wv[i] = alpha * xv[i] + yv[i];
       });
     } else {
       // w = alpha * x + beta * y
       Kokkos::parallel_for(n, KOKKOS_LAMBDA (const local_int_t i) {
         wv[i] = alpha * xv[i] + beta * yv[i];
       });
     }
     return 0;
   }
```

VSCode with GitHub Copilot Plugin

- Same approach for Kokkos

VSCode+Copilot
HPCG Benchmark WAXPBY Kernel

Kokkos_ComputeWAXPBY.cpp 9+, U

src > Kokkos_ComputeWAXPBY.cpp > ComputeWAXPBY(const local_int_t, const double, const Vec

```cpp
// generate computeWAXPBY function using Kokkos::view and Kokkos::parallel_for
int ComputeWAXPBY(const local_int_t n, const double alpha, const Vector & x,
    const double beta, const Vector & y, Vector & w) {

  assert(x.localLength>=n); // Test vector lengths
  assert(y.localLength>=n);

  const double * const xv = x.values;
  const double * const yv = y.values;
  double * const wv = w.values;

  if (alpha==1.0) {
    if (beta==1.0) {
      // w = x + y
      Kokkos::parallel_for(n, KOKKOS_LAMBDA (const local_int_t i) {
        wv[i] = xv[i] + yv[i];
      });
    } else {
      // w = x + beta * y
      Kokkos::parallel_for(n, KOKKOS_LAMBDA (const local_int_t i) {
        wv[i] = xv[i] + beta * yv[i];
      });
    }
  } else if (beta==1.0) {
    // w = alpha * x + y
    Kokkos::parallel_for(n, KOKKOS_LAMBDA (const local_int_t i) {
      wv[i] = alpha * xv[i] + yv[i];
    });
  } else {
    // w = alpha * x + beta * y
    Kokkos::parallel_for(n, KOKKOS_LAMBDA (const local_int_t i) {
      wv[i] = alpha * xv[i] + beta * yv[i];
    });
  }
  return 0;
}
```

Apple_ComputeWAXPBY_ref.cpp 9+, U | Intel_ComputeWAXPBY.cpp 9+, U | ComputeW

src > Apple_ComputeWAXPBY_ref.cpp

```cpp
// generate computeWAXPBY_ref function using Apple vector intrinsics:
int ComputeWAXPBY_ref(const local_int_t n, const double alpha, const Vector & x,
    const double beta, const Vector & y, Vector & w) {

  assert(x.localLength>=n); // Test vector lengths
  assert(y.localLength>=n);

  const double * const xv = x.values;
  const double * const yv = y.values;
  double * const wv = w.values;

  if (alpha==1.0) {
    if (beta==1.0) {
      // w = x + y
      for (local_int_t i=0; i<n; i+=4) {
        __builtin_prefetch(&xv[i+16], 0, 0);
        __builtin_prefetch(&yv[i+16], 0, 0);
        __builtin_prefetch(&wv[i+16], 1, 1);
        const vector double vx = vec_ld(0, &xv[i]);
        const vector double vy = vec_ld(0, &yv[i]);
        const vector double vw = vec_add(vx, vy);
        vec_st(vw, 0, &wv[i]);
      }
    } else {
      // w = x + beta * y
      const vector double vb = vec_splats(beta);
      for (local_int_t i=0; i<n; i+=4) {
        __builtin_prefetch(&xv[i+16], 0, 0);
        __builtin_prefetch(&yv[i+16], 0, 0);
        __builtin_prefetch(&wv[i+16], 1, 1);
        const vector double vx = vec_ld(0, &xv[i]);
        const vector double vy = vec_ld(0, &yv[i]);
        const vector double vw = vec_madd(vb, vy, vx);
        vec_st(vw, 0, &wv[i]);
      }
    }
```

VSCode+Copilot
HPCG Benchmark WAXPBY Kernel

VSCode with GitHub Copilot Plugin

- Same approach for Apple ISA
- And so on

VSCode and Copilot for LaTeX

- VSCode with:
  - LaTeX Workshop plugin
  - GitHub Copilot plugin
  - Grammerly plugin
- Makes an awesome writing environment:
  - Syntax-aware
  - Text predicting – including LaTeX syntax
  - Bug-free compilation
  - And much more..

OpenAI Playground
AztecOO Belos Parameters

Help | Personal

Playground

Load a preset... | Save | View code | Share

Build a teuchos parameterlist to solve an ill-conditioned non-symmetric linear system using muelu and belos

```
Teuchos::ParameterList MueLuList;

// MueLu preconditioner parameters
MueLuList.set("max levels", 10);
MueLuList.set("prec type", "MGV");
MueLuList.set("aggregation: type", "uncoupled");
MueLuList.set("aggregation: damping factor", 0.5);
MueLuList.set("aggregation: max agg size", 3);
MueLuList.set("aggregation: min agg size", 2);
MueLuList.set("smoother: type", "Chebyshev");
MueLuList.set("smoother: sweeps", 5);
MueLuList.set("smoother: pre or post", "both");
MueLuList.set("coarse: type", "Amesos-KLU");

// Belos solver parameters
Teuchos::ParameterList BelosList;
BelosList.set("Maximum Iterations", 500);
BelosList.set("Convergence Tolerance", 1e-10);
BelosList.set("Verbosity", 33);
BelosList.set("Output Frequency", 10);
BelosList.set("Output Style", 1);
BelosList.set("Implicit Residual Scaling", "Norm of Initial Residual");

// Combine MueLu and Belos parameter lists
Teuchos::ParameterList params;
params.set("Preconditioner", MueLuList);
params.set("Solver", BelosList);
```

OpenAI Playground
BibTeX citations
Convert from MLA-style format

Convert the following citations to bibtex format:
Articles:
U. M. Yang, L. C. McInnes, "xSDK: Building an ecosystem of highly efficient math libraries for exascale", SIAM News v. 54 issue 1, pages 8-9, January 2021.
P. Luszczek, U. Yang, "Building Community through Software Policies", Better Scientific Software, blog article, https://bssw.io/blog_posts/building-community-through-software-policies, August 2019

Plenary talks:
Ulrike Yang, "Elements of an Effective Ecosystem of Math Libraries for Exascale", invited plenary talk at Second Joint SIAM/CAIMS Annual Meeting (AN20), July 6-17, 2020, online.
Ulrike Yang, "xSDK: an Ecosystem of Interoperable Independently Developed Math Libraries", invited plenary talk at 1st MarDI Workshop on Scientific Computing, Muenster, Germany, Oct 26-28, 2022.

Invited presentations:
U. M. Yang, "Performance Portability in the Extreme-scale Scientific Software Kit", invited talk, ECCOMAS 2022, June 3-6, 2022, Oslo, Norway.
U. M. Yang, "Building an Effective Ecosystem of Math Libraries for Exascale", invited talk, E4S Workshop, EuroMPI/US'20, 9-24-20
U. M. Yang, P. Lusczcek, "Building an Effective Ecosystem with Community Policies", invited talk, SIAM Conference on Computational Science and Engineering (SIAM CSE21), March 1-5, 2020, virtual.
U. M. Yang, "xSDK: a Community of Diverse HPC Software Packages", invited talk, The First Extreme-scale Scientific Software Forum (E4S Forum), IEEE Cluster 2019, Albuquerque NM, 9-23-19
Ulrike Yang, "xSDK: a Community of Diverse Numerical HPC Software Packages", invited talk, Final SPPEXA Symposium, October 21-23, 2019, Dresden, Germany.
Ulrike Yang, "xSDK: Foundations of a Numerical Software Ecosystem for High-performance CSE", SIAM CSE 2019, Feb 25 – Mar 1, 2019, Spokane, invited talk.

Minisymposia:
Co-organized 2-part minisymposium at SIAM CSE21: K. Teranishi, U. Yang, J. Willenbring, "Programming Environments for Exascale Computing – Ecosystem Perspectives", SIAM Conference on Computational Science and Engineering (SIAM CSE21), March 1-5, 2020,

ECP annual meeting posters and sessions:
"Extreme-scale Scientific Software Development Kit (xSDK) for ECP", poster at 2020 ECP annual meeting, Houston, TX, Feb 5, 2020.
"Extreme-scale Scientific Software Development Kit (xSDK) for ECP", poster at 2021 ECP annual meeting, virtual, April 12-16, 2021.
"Extreme-scale Scientific Software Development Kit (xSDK) for ECP", poster at 2022 ECP annual meeting, virtual, May 2022.
T. Munson, J. Willenbring, U. M. Yang, "What can be learned from applying `team of teams' principles to Exascale Computing Projects PETSc, Trilinos, xSDK and E4S?", Breakout Session, ECP Annual Meeting 2022, May 4,2022.

OpenAI Playground
BibTeX citations
To BibTeX database format

```
@article{yang2021xsdk,
  title={xSDK: Building an ecosystem of highly efficient math libraries for exascale},
  author={U. M. Yang and L. C. McInnes},
  journal={SIAM News},
  volume={54},
  number={1},
  pages={8-9},
  year={2021}
}

@misc{yang2019building,
  title={Building Community through Software Policies},
  author={P. Luszczek and U. Yang},
  howpublished={\url{https://bssw.io/blog_posts/building-community-through-software-policies}},
  year={2019}
}

@inproceedings{yang2020elements,
  title={Elements of an Effective Ecosystem of Math Libraries for Exascale},
  author={U. Yang},
  booktitle={Second Joint SIAM/CAIMS Annual Meeting (AN20)},
  year={2020},
  url={https://www.siam.org/meetings/an20/}
}

@inproceedings{yang2022xsdk,
  title={xSDK: an Ecosystem of Interoperable Independently Developed Math Libraries},
  author={U. Yang},
  booktitle={1st MarDI Workshop on Scientific Computing},
  year={2022},
  address={Muenster, Germany}
}

@inproceedings{yang2022performance,
  title={Performance Portability in the Extreme-scale Scientific Software Kit},
  author={U. M. Yang},
  booktitle={ECCOMAS 2022},
  year={2022},
  address={Oslo, Norway}
}
```

```
@inproceedings{yang2020building,
  title={Building an Effective Ecosystem of Math Libraries for Exascale},
  author={U. M. Yang},
  booktitle={E4S Workshop, EuroMPI/US'20},
  year={2020},
  url={https://www.euro-mpi.org/2020/}
}

@inproceedings{yang2020xsdk,
  title={xSDK: a Community of Diverse HPC Software Packages},
  author={U. M. Yang},
  booktitle={The First Extreme-scale Scientific Software Forum (E4S Forum), IEEE Cluster 2019},
  year={2019},
  address={Albuquerque NM}
}

@inproceedings{yang2019xsdk,
  title={xSDK: a Community of Diverse Numerical HPC Software Packages},
  author={U. Yang},
  booktitle={Final SPPEXA Symposium},
  year={2019},
  address={Dresden, Germany}
}
```

OpenAI Playground
Tl;dr assistant

Tl;dr. Here is a WWW'04 conference paper abstract I use in a compact writing exercise, followed by the AI-generated Tl;dr version. I chose the abstract because it presents significant opportunities for improvement.

Original text:

> We seek to gain improved insight into how Web search engines should cope with the evolving Web, in an attempt to provide users with the most up-to-date results possible. For this purpose we collected weekly snapshots of some 150 Web sites over the course of one year, and measured the evolution of content and link structure. Our measurements focus on aspects of potential interest to search engine designers: the evolution of link structure over time, the rate of creation of new pages and new distinct content on the Web, and the rate of change of the content of existing pages under search-centric measures of degree of change. Our findings indicate a rapid turnover rate of Web pages, i.e., high rates of birth and death, coupled with an even higher rate of turnover in the hyperlinks that connect them. For pages that persist over time we found that, perhaps surprisingly, the degree of content shift as measured using TF.IDF cosine distance does not appear to be consistently correlated with the frequency of content updating. Despite this apparent non-correlation, the rate of content shift of a given page is likely to remain consistent over time. That is, pages that change a great deal in one week will likely change by a similarly large degree in the following week. Conversely, pages that experience little change will continue to experience little change. We conclude the paper with a discussion of the potential implications of our results for the design of effective Web search engines.

Tl;dr version

> The Web is constantly evolving, with new pages being created and old ones being deleted or modified. Search engines need to be able to cope with this changing landscape in order to provide users with the most up-to-date results possible. Our study found that the rate of change of Web content is much higher than the rate of change of the links that connect it. This means that search engines need to be able to adapt quickly to changes in the content of Web pages, in order to provide the most relevant and up-to-date results.

OpenAI Playground
Second-grader

Here is an opening paragraph for an article I wrote a few years ago, followed by the second-grader translation. Very interesting that the AI inserted the definition of exascale computing.

Prompt:

Summarize this (my paragraph) for a second-grade student:

The demands of massive concurrency and increased resilience required for effective exascale computing have led to claims that our existing approach to application programming must be replaced. In this article I argue that disruption is inevitable, but from the programming perspective will not require new languages or programming environments. Instead the disruption will be in application architecture, introducing a new control layer that will provide the concurrency, adaptability and resilience features we need in order to achieve effective exascale performance levels.

OpenAI Playground Generated Second-grader text:

The article is discussing how the way we currently write programs will have to change in order to be effective at exascale computing. Exascale computing is a type of computing where programs can use a billion billion calculations per second. This is different from our current approach, which the article claims will need to be replaced in order to be effective. The new way of writing programs will include a new control layer that provides features like concurrency, adaptability, and resilience.

Writing detailed prompts help to produce more usable content

OpenAI Playground
More specific prompts

Playground

Load a preset...

Save

Using an existing 2-slide presentation, with bullet points on the second slide, write a python script to update the bullet points on the second slide

```
#Import the necessary libraries

from pptx import Presentation

#Open the presentation

prs = Presentation('existing_2_slide_presentation.pptx')

#Update the bullet points on the second slide

for slide in prs.slides:
    if slide.slide_index == 1:
        for shape in slide.shapes:
            if shape.has_text_frame:
                for paragraph in shape.text_frame.paragraphs:
                    for run in paragraph.runs:
                        run.text = 'Updated bullet point'

#Save the updated presentation

prs.save('updated_2_slide_presentation.pptx')
```

Takeaways – AI Tools Transform our ability to use software APIs

- **Script APIs:** Makes possible the use of GitHub API example:
  - Go from no knowledge to a working meta-data access script in 15 minutes (with help from StackOverflow)
- **Device-specific APIs:** WAXPBY generic C++ kernel:
  - Generate starter versions for Intel/Apple vector ISA, Kokkos, and more
  - Accelerates transition to Kokkos
  - Perhaps increases tolerance for code divergence?
- **Scientific library APIs:** Trilinos (AztecOO, Ifpack, Muelu, Belos)
  - Generate starter parameter lists
  - Generate library calls
  - Can change how we ramp up new user

Takeaways – Makes writing better, faster, cheaper

- BibTeX – Easy tedious work
- Predictive, syntax-aware text generation (LaTeX)
- Compact writing – Summarize/Tl;dr
- Translation for different audiences – 2nd grader
- Companion for all writing activities

Takeaways – Generative AI Tools give us (way) better, faster, and cheaper

- We need to assure that everyone who generates or transforms content has access to generative tools
- Sooner rather than later – stakes are high!
- Two paths to explore for isolated environments:
  - Drop GPTX.Y on a hard drive and bring it in-house
  - Internally build models using custom-curated content
  - Both paths seem essential