Essential Building Blocks for Creating an Open-source EDA Project

2019 IEEE/ACM Design Automation Conference (DAC)

June 4th | Las Vegas, NV

Tsung-Wei Huang, Chun-Xun Lin, Guannan Guo, and Martin Wong
University of Illinois at Urbana-Champaign (UIUC), IL

The EDA/CAD Research Landscape

- **My first DAC paper …**
  - New algorithm
  - New better results
  - Benchmarks & testcases
  - Internal prototype code

New problem formulation in …

New algorithm and implementation to outperform existing solutions by …

Experimental results showed …

Progressive Network-Flow Based Power-Aware Broadcast Addressing for Pin-Constrained Digital Microfluidic Biochips

Tsung-Wei Huang, Hong-Yan Su, and Tsung-Yi Ho*
Department of Computer Science and Information Engineering
National Cheng Kung University, Tainan, Taiwan
twhuang@eda.csie.ncku.edu.tw; lion78814@gmail.com; tyho@csie.ncku.edu.tw

ABSTRACT

In recent emerging marketplace, designs for pin-constrained digital microfluidic biochips (PDMFBs) have received much attention due to the large impact on packaging and product cost. One of the major approaches, *broadcast addressing*, reduces the pin count by assigning a single control pin to multiple electrodes with mutually-compatible control signals. Prior works utilize this addressing scheme by minimally grouping electrode sets with non-conflict signal merging. However, merging control signals also introduces redundant actuations, which potentially cause a high power-consumption problem. Recent studies on PDMFBs have indicated that high power consumption not only decreases the product lifetime but also degrades the system reliability. Unfortunately, this power-aware design concern is still not readily available among current design automations of PDMFBs. To cope with these issues, we propose in this paper the *first* power-aware broadcast addressing for PDMFBs. Our algorithm simultaneously takes pin-count reduction and power-consumption minimization into consideration, thereby achieving higher integration and better design performance. Experimental results demonstrate the effectiveness of our algorithm.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithms, Performance, Design

Keywords

Digital microfluidics, electrode addressing, power

1. INTRODUCTION

As the microfluidic technology advances, digital microfluidic biochips (DMFBs) have attracted much attention recently. These miniaturized and automated DMFBs provide various advantages including high portability, high throughput, high sensitivity, high immunity to human intervention, and low sample volume consumption. Due to these advantages, more and more practical applications such as infant health care, point-of-care disease diagnostics, environmental toxin monitoring, and drug discovery have been successfully realized on DMFBs [2, 5, 11].

*This work was partially supported by the National Science Council of Taiwan ROC under Grant No. NSC 99-2220-E-006-005 and 99-2221-E-006-220.

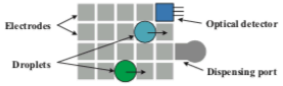


Figure 1: Schematic view of a digital microfluidic biochip.

Typically, a DMFB consists of a two dimensional (2D) electrode array, optical detector, and dispensing port, as schematically shown in Figure 1 [5]. In performing fluidic-handling functions, droplet-based operations are introduced on DMFB platforms. By generating electrohydrodynamic forces from electrodes, droplets can be dispensed from dispensing ports, moved around the 2D array for performing reactions (e.g., mixing or dilution), and then moved toward the optical detector for detection [10]. The entire operations are also called *reconfigurable* operations due to their flexibility in area and time domain [1].

In realizing fluidic controls, a primary issue is the control scheme of electrodes. To correctly control the electrodes, *electrode addressing* is introduced as a method through which electrodes are assigned by control pins to identify input signals. Early DMFB designs relied on *direct addressing*, where each electrode is directly and *independently* assigned by a dedicated control pin [4], as illustrated in Figure 2(a). This addressing maximizes the flexibility of electrode controls. However, for large arrays, the high pin-count demand complicates the electrical connections between the chip and the external controller, thus rendering this kind of chip unreliable and prohibitively expensive to package and manufacture [4, 5, 13, 14].

Recently, *pin-constrained* DMFBs (PDMFBs) have raised active discussions to overcome this problem. One of the major approaches, *broadcast addressing*, provides high throughput for bioassays and reduces the number of control pins by identifying and connecting them with *compatible* control signals. In other words, multiple electrodes are controlled by a single signal source and are thus actuated simultaneously, as shown in Figure 2 (b). In this regard, much on-going effort has been made to group sets of electrodes that can be driven uniformly without introducing any signal conflict [9, 13, 14].

2011 IEEE/ACM DAC, San Diego, CA

A Critical Question

- ***How does the community benefit from reading this?***
  - ☺ Presented a new problem formulation
  - ☺ Presented a new algorithm and implementation
  - ☺ Presented large improvement over existing solutions
  - ☹ Performance evaluation is "selective"
  - ☹ Difficult to "reproduce" the result
  - ☹ Wasted time on "re-implementing" the code

We want new algorithms & results:

- Open and accessible
- Fully reproducible
- Easy to integrate to my packages
- Ready to use/alter by other scientists

Why Are We Sluggishly Changing this?

- **From the academic perspective …**
  - effort (prototype code) << effort (production code)
  - Does not reward software/system development
  - Promotion is largely based on scientific papers
  - Slow acceptance of the scientific software engineer
- **From the industrial perspective … ***
  - cost (software error) << cost (hardware error)
  - Wants to keep algorithms/IPs confidential
  - Tools are highly customer-driven, lacking API standards
  - The monopoly locks people to proprietary tools

Extremely inefficient and unsatisfying!

** Conversation with our industrial partners in EDA/CAD companies*

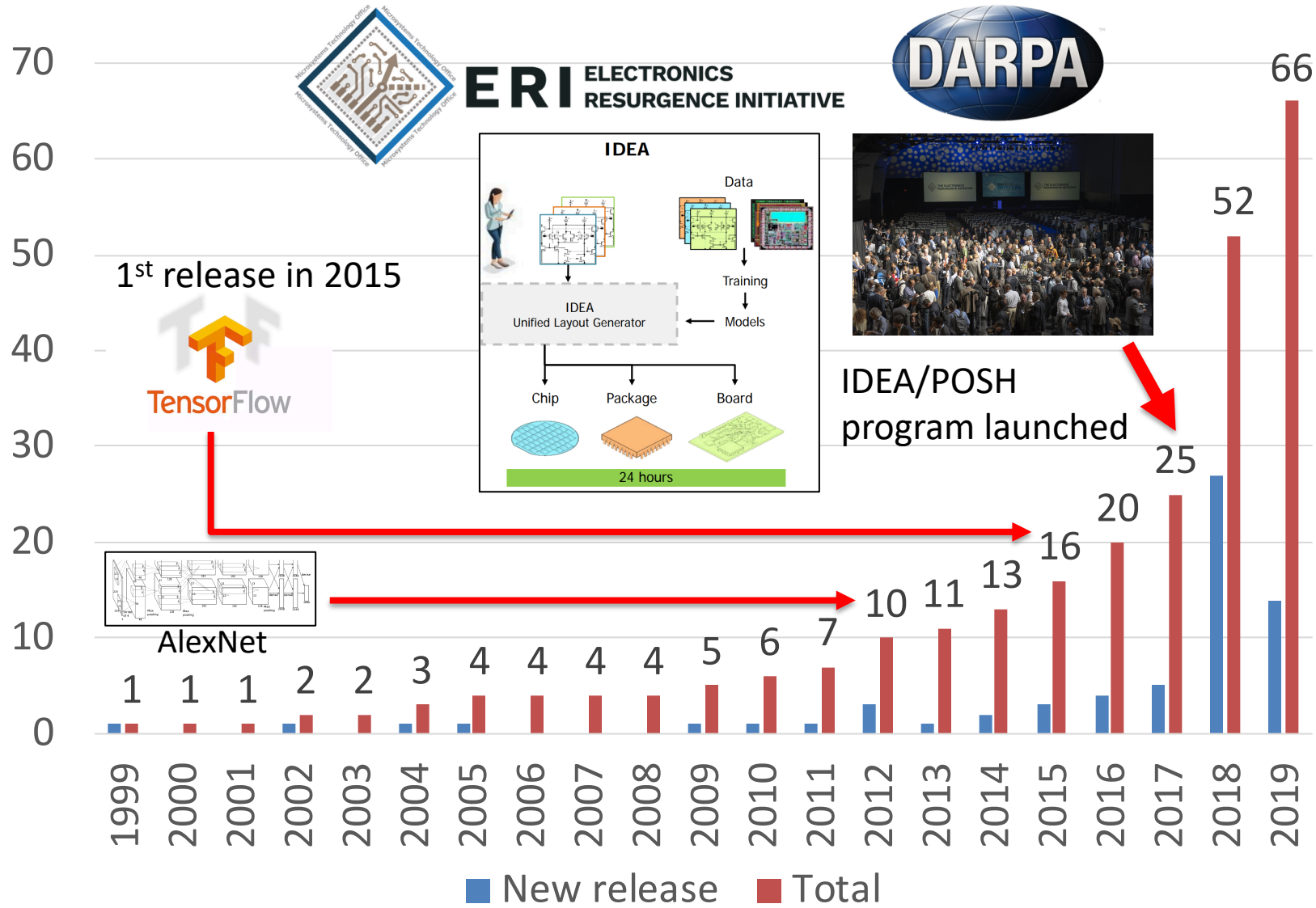
The Most Essential Building Block: Mindset

- **Let's work together to change the system**
  - Open source to enable quick sharing of new ideas
- **Publication systems should credit software dev**
  - Innovation should include system implementation
    - API, software architecture, documentation, design strategies
  - Artifact reproducibility evaluation using ACM badges

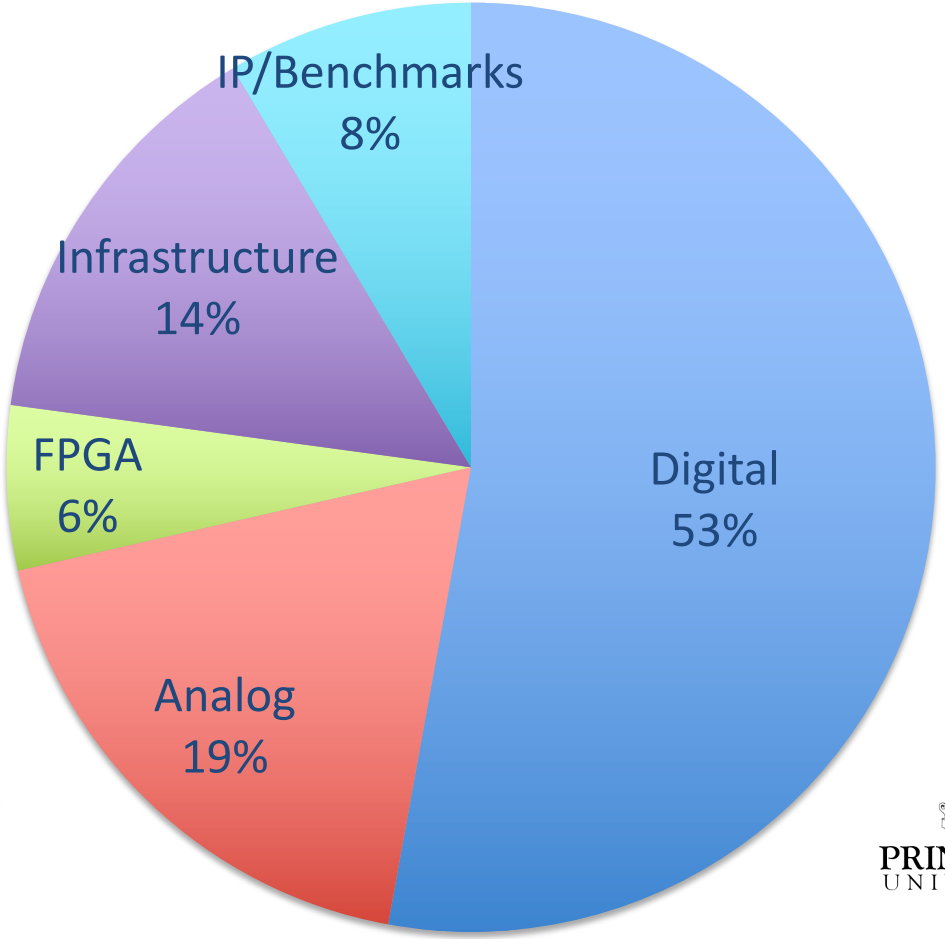
ACM artifact review and badging
https://www.acm.org/publications/policies/artifact-review-badging

Let's go even further
- 57th DAC will include 30% tool papers
- Code review as a main judge
- TPC will include code reviewers
- Software patches are contributions
- 1st ACM/IEEE SysCAD conference

Open-Source EDA Projects Activities

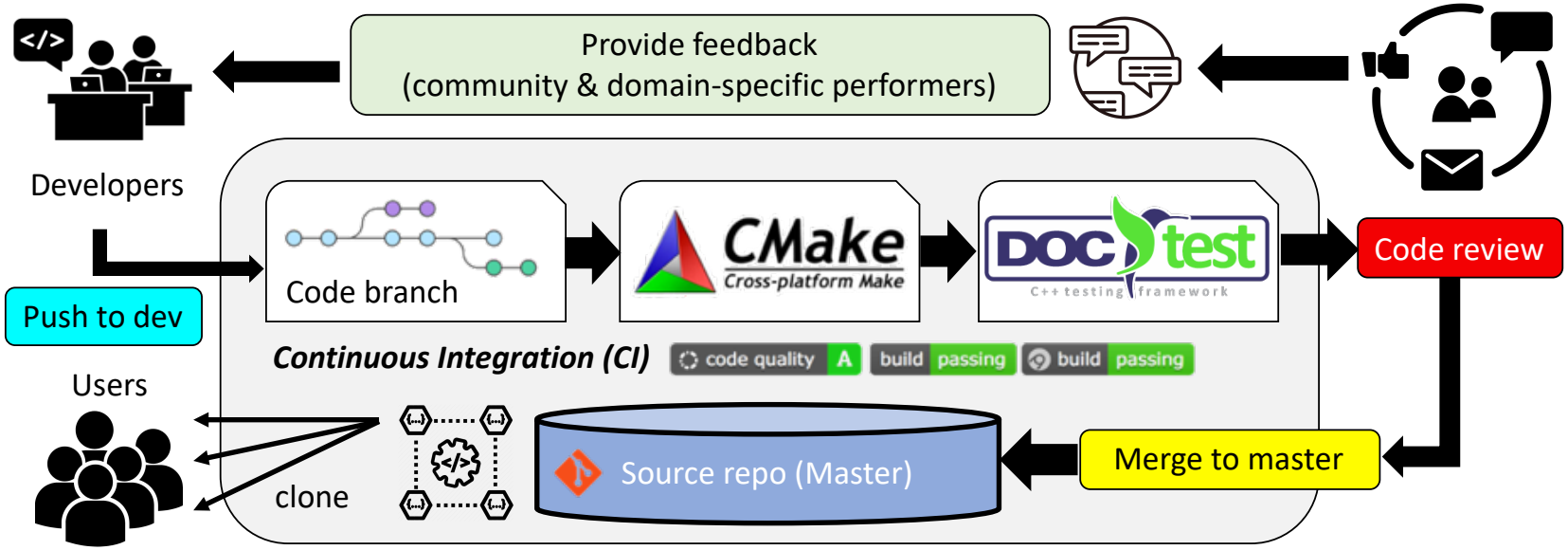


What do these Projects Do?



Most projects are from academia …

A Healthy Open-source Development Cycle



- ❑ Understand your users and what you are doing
- ❑ Things to know in creating a repository
- ❑ Prepare an informative README and documentation
- ❑ Set up a contribution guideline
- ❑ Iterate the feedback loop

Understand What Your Users Need

- **Roughly speaking**
  - Developers take your project to do “derived” work
    - For example, a parallel programming library
  - End-users take your project to do “standalone” work
    - For example, a C++ debugger or a performance profiler

Developers are normally respectful

Talk technically;
Care API and reference;
Write code and software;

Talk generally;
Care doc and usability;
Use software and tools;

End-users are often friendly …

Open-source owners can be both developers and end-users, but it’s important to understand the target users of your projects

Code of Conduct

- **It is a free world, especially in open source**
  - You cannot force others to use your tools
  - No ones owe you to use your tools
- **Put respect to the highest standards**
  - Nobody is ever going to be the top coder in the world
  - Open source means open collaboration
    - Minimize risk, shared effort, quick prototyping
  - Respect users' need and their intent
  - Respect opportunities and opponents

Google

"Don't be evil"

Never ignore the importance of respect even though the project is free

Things to Know in Creating a Repository

- **A repository helps store and manage code with**
  - Git version control (branch capabilities)
  - Cloud-based service (GitHub, Bitbucket, GitLab)
  - Issue tracker, open forum, contribution environment
- **Name your project wisely**
  - Precise, specific, no jargon
  - Keep the name to be 7-10 words
- **Tag your project to the right search categories**
  - Language, functionality, algorithm, library
- **Attach a proper license to your project**

Example: Cpp-Taskflow's Front Page

cpp-taskflow / cpp-taskflow

Unwatch 125 | Unstar 1,752 | Fork 182

Code | Issues 9 | Pull requests 1 | Security | Insights | Settings

Modern C++ Parallel Task Programming Library https://cpp-taskflow.github.io

Edit

taskflow | task-based-programming | cpp17 | parallel-programming | threadpool | concurrent-programming | header-only | flowgraph | high-performance-computing | multicore-programming | multi-threading | taskparallelism | multithreading | parallel-computing | work-stealing | scheduling-algorithms | scheduler

Manage topics

865 commits | 2 branches | 2 releases | 1 environment | 14 contributors | View license

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

tsung-wei-huang updated executor

Latest commit a1eb7c0 4 days ago

Manage the topic tags

License

Project name
(6 specific words)

Default branch (master)

Project activities
(keep your project active by at least one commit in 3 days)

Similar ideas apply to other platforms (GitLab, Bitbucket) as well.

Cpp-Taskflow: https://github.com/cpp-taskflow/cpp-taskflow

Comparison of Popular Licenses

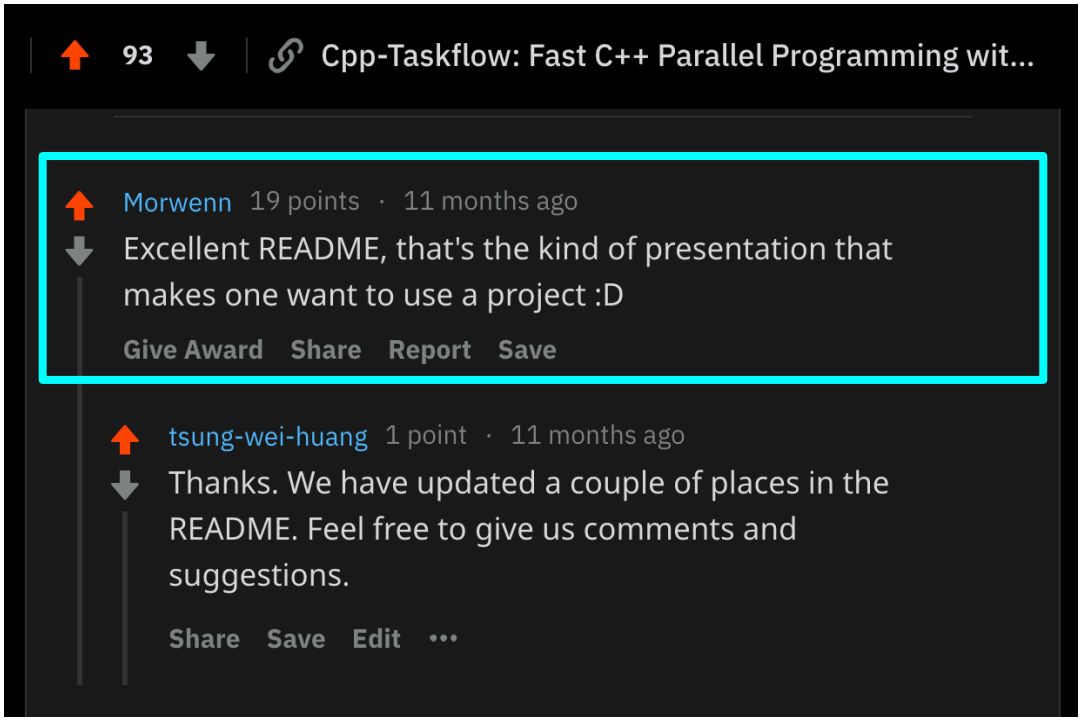
Terms and Use		GNU GPLv3	Apache License 2	MIT License
Permissions	Commercial use	✓	✓	✓
	Distribution	✓	✓	✓
	Modification	✓	✓	✓
	Patent use	✓	✓	
	Private use	✓	✓	✓
Conditions	Disclose source	✓		
	License & copyright	✓	✓	✓
	Same license	✓		
	State changes	✓	✓	
Limitations	Liability	✓	✓	✓
	Trademark use		✓	
	Warranty	✓	✓	✓

In a nutshell, the main difference between Licenses is on the restriction of "derived" work and its "redistribution".

Open-source License: https://choosealicense.com/licenses/

Prepare for an Effective README

- The **most important** component in your project



Keep in mind thousands of projects are being created everyday; the majority of people glance and leave.

HOOK YOUR USER

Points to take care:

- What/Why/Where
- Code example
- Installation guide
- System environment
- Doc & API reference
- Reward contributors

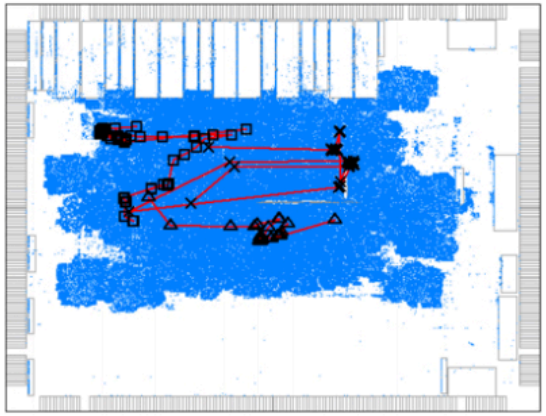
OpenTimer

build passing | c++ 17 | download latest | version v2.0-alpha | Ask me anything | doc wiki | License MIT

A High-Performance Timing Analysis Tool for VLSI Systems

Why OpenTimer?

OpenTimer is a new static timing analysis (STA) tool to help IC designers quickly verify the circuit timing. It is developed completely from the ground up using *C++17* to efficiently support parallel and incremental timing. Key features are:



- Industry standard format (.lib, .v, .spef, .sdc) support
- Graph- and path-based timing analysis
- Parallel incremental timing for fast timing closure
- Award-winning tools and golden timers in CAD Contests

Get Started with OpenTimer

The easiest way to start using OpenTimer is to use *OpenTimer shell*. OpenTimer shell is a powerful tool for interactive analysis and the simplest way to learn core functionalities. Compile OpenTimer and launch the shell program `ot-shell` under the `bin` directory.

OpenTimer: https://github.com/OpenTimer/OpenTimer

Document your Project

- **As important as other development facets**
  - Reminds you of what you code
  - Reduce users' time spent on understanding your code
- **But, what is the problem?**
  - The main reason code goes undocumented is time
  - Code abstraction happens before documentation

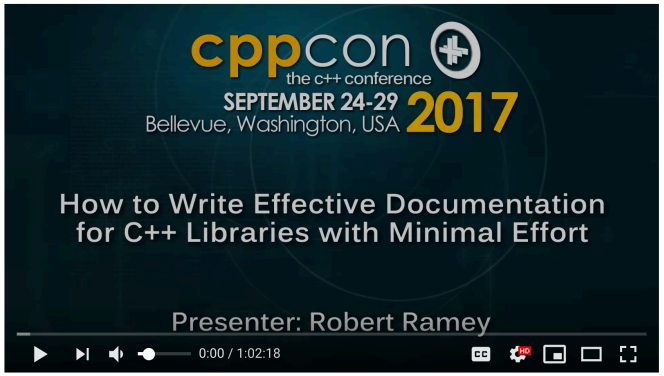
"An incredible 93% of people reported being frustrated with incomplete or confusing documentation," Robert Ramey

- **A suggested solution**
  - Craft code and documentation together (e.g., Doxygen)

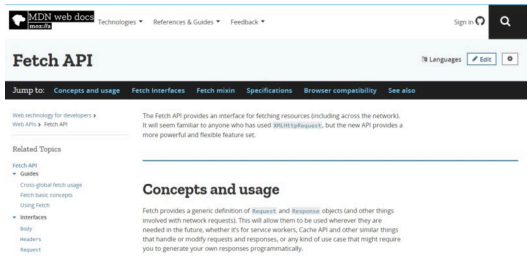
"If you spent 6 hours on writing code, spend at least another 6 hours on documenting your code," C++ Conference Keynote

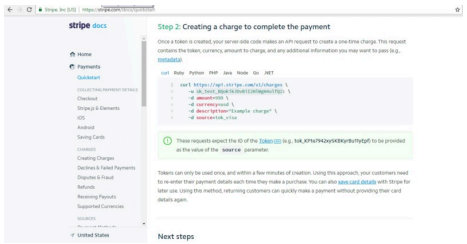
Resources to Document Your Code

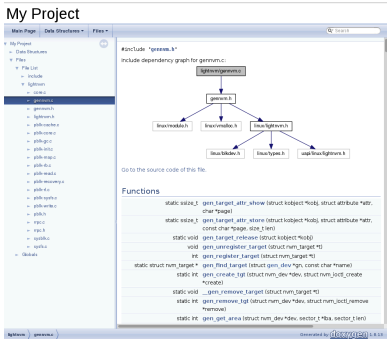
- **Good code does need good documentation**
  - Never forego the need of doc
- **Some popular examples**
  - MDN
  - Dijango
  - Stripe
  - Doxygen
- **My personal taste**
  - C++ reference
  - Boost documentation



CppCon 2017: Robert Ramey "How to Write Effective Documentation for C++ Libraries..."



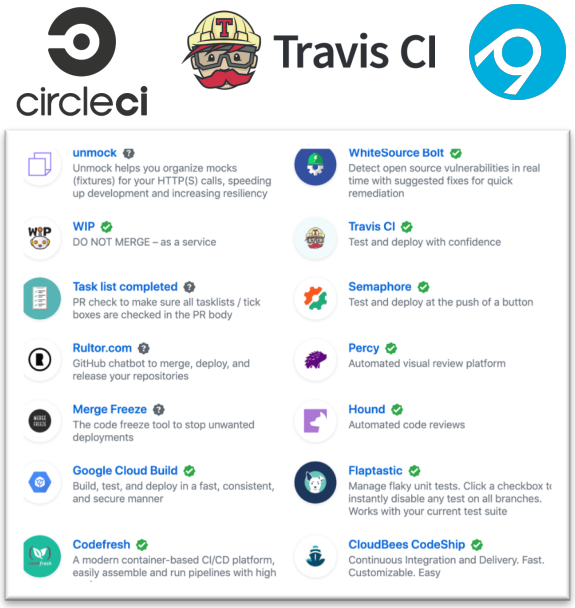




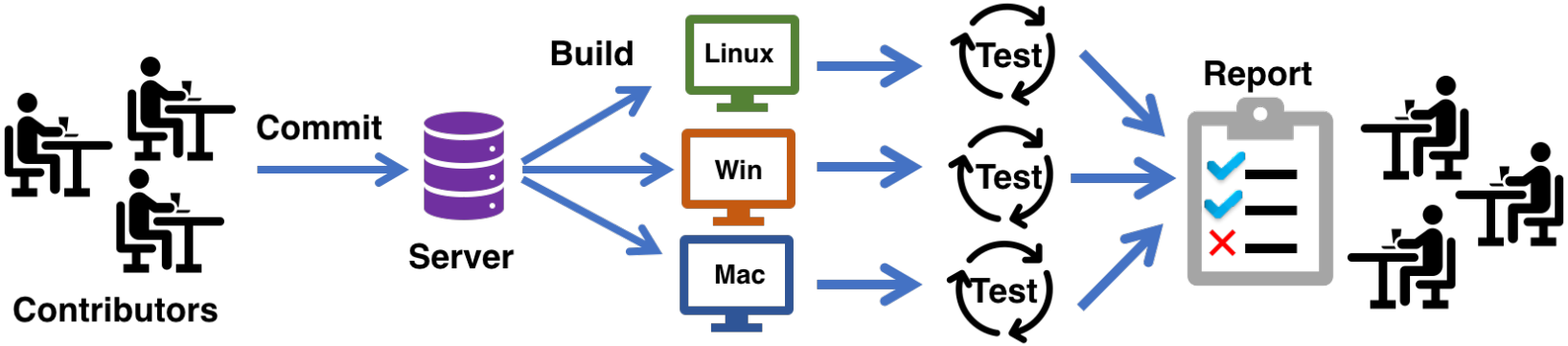
"If you write good documentation, most likely you will write a good scientific paper," my manager at Citadel

Grow your Project Community

- **Attract people to contribute**
  - Turn end-users to developers
  - Getting pull requests is not easy
  - Proof of your project creditability
- **A good contribution environment**
  - Template, code review, refactor
  - Continuous integration
    - Ensure each change doesn't break



Continuous integration tools



Current | Branches | Build History | Pull Requests | More options

✓ **master** updated executor

Commit `a1eb7c0`
Compare `c7abd3d..a1eb7c0`
Branch `master`

Tsung-Wei Huang

#662 passed

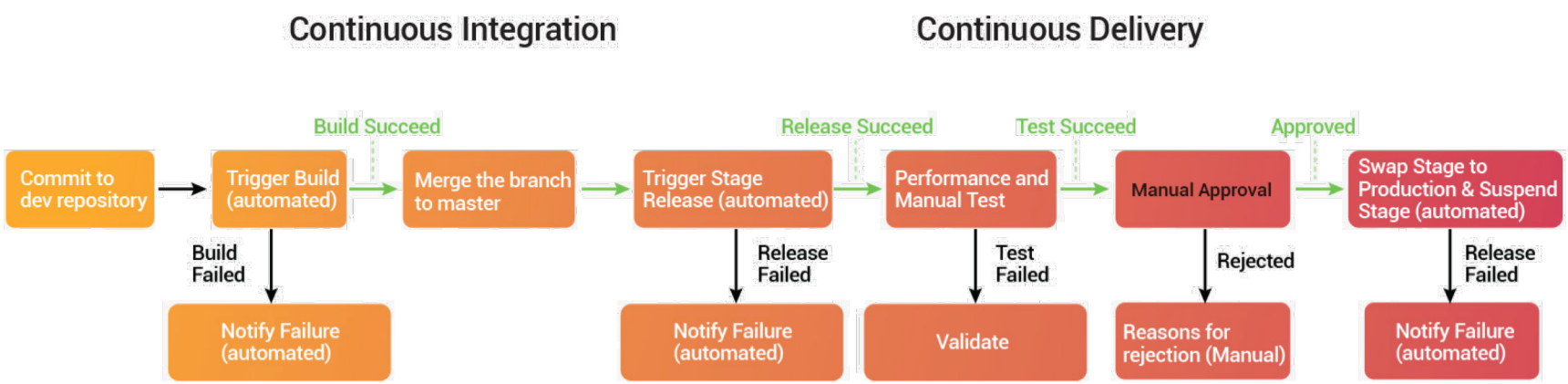
Ran for 6 min 55 sec
Total time 13 min 18 sec

4 days ago

target (g++, clang, etc.)

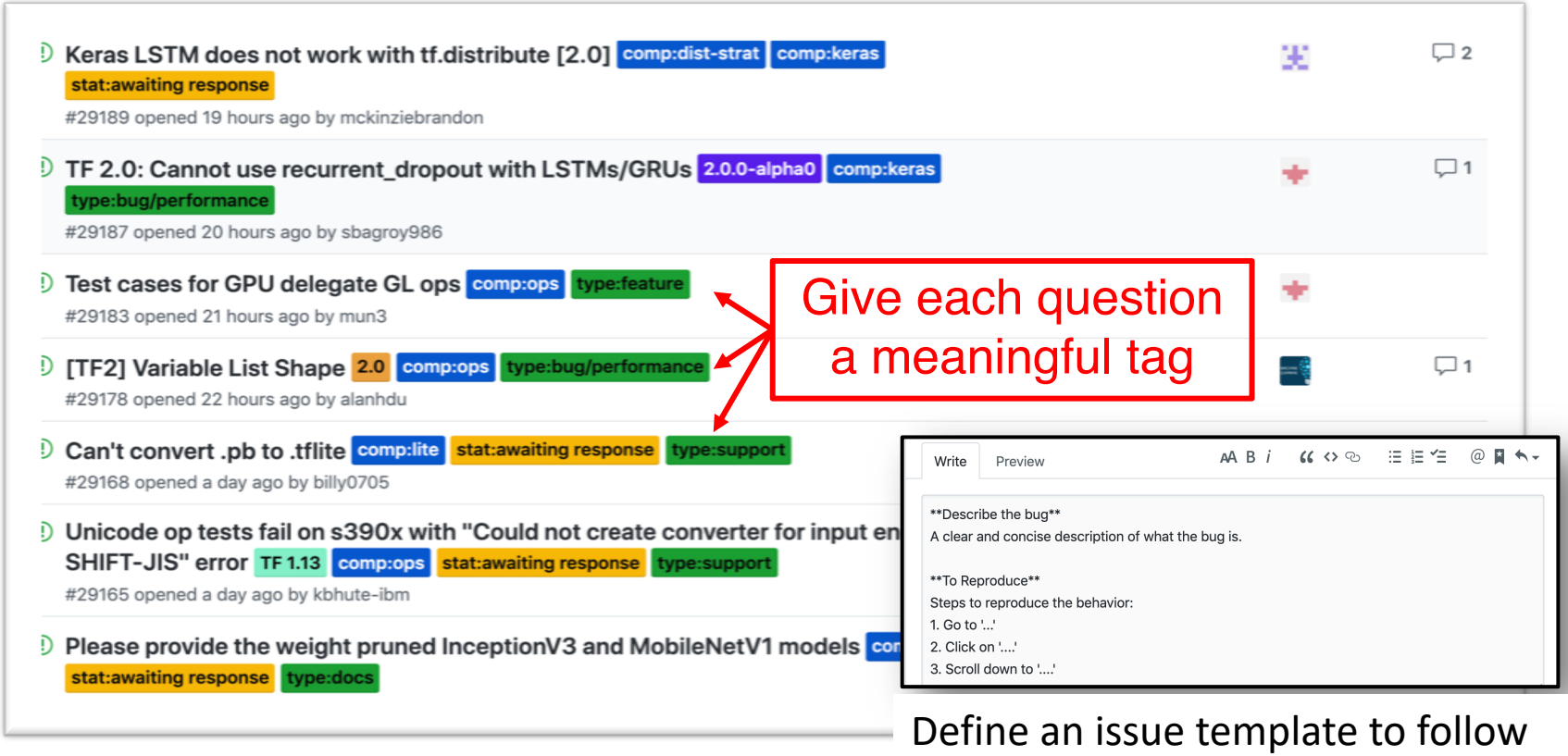
Build jobs | View config

✓	# 662.1	Compiler: g++ C++	MATRIX_EVAL="CC=gcc-7 && CXX=g++-7"	3 min 13 sec
✓	# 662.2	Compiler: g++ C++	MATRIX_EVAL="CC=gcc-8 && CXX=g++-8"	2 min 58 sec
✓	# 662.3	Compiler: clang++ C++	MATRIX_EVAL="CC=clang-6.0 && CXX=clang++-6.	3 min 37 sec
✓	# 662.4	Compiler: clang++ C++	MATRIX_EVAL="CC=clang-7 && CXX=clang++-7"	3 min 30 sec



Iterate the Feedback Loop

- **Use feedback to improve the project**
  - Communicate with users through an open forum
  - Manage your forum effectively



Define an issue template to follow

Block-interleaved block storage in block-Jacobi #159

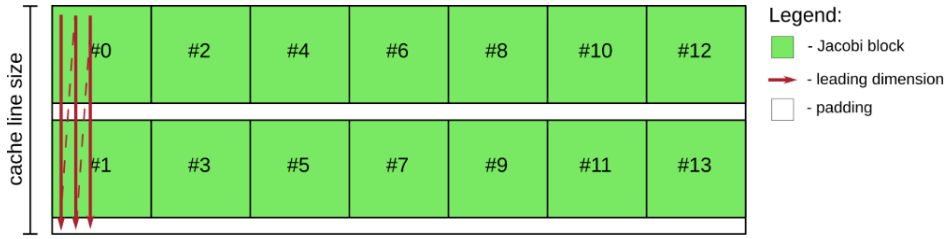
Merged gflegar merged 3 commits into `develop` from `interleaved_block_jacobi` on Nov 26, 2018

Conversation 10 | Commits 3 | Checks 0 | Files changed 9

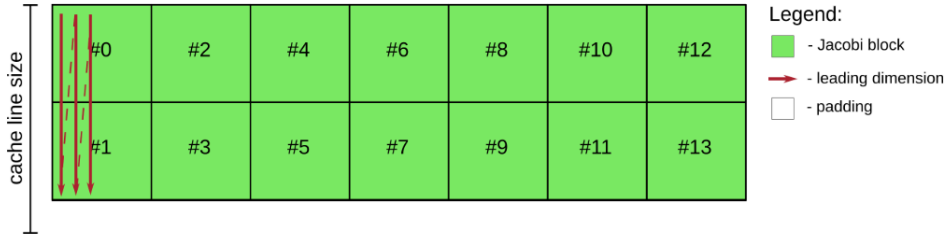
gflegar commented on Oct 31, 2018 • edited — Member

This PR further improves the performance of the block-Jacobi preconditioner for smaller block sizes by redesigning the way blocks are stored in memory. In addition to column-major storage introduced in #158, this PR interleaves the blocks to maximize coalescence when a single warp handles multiple problems.
The idea is shown in the following figure, where the maximum block size allows to interleave 2 blocks to fill the cache line:

Option 1:



Option 2:



There's trade-off in both approaches depicted in the figure. Option 1 always results in aligned data access, but consumes more memory in total. Option 2 consumes less memory, but data accesses are not always aligned.

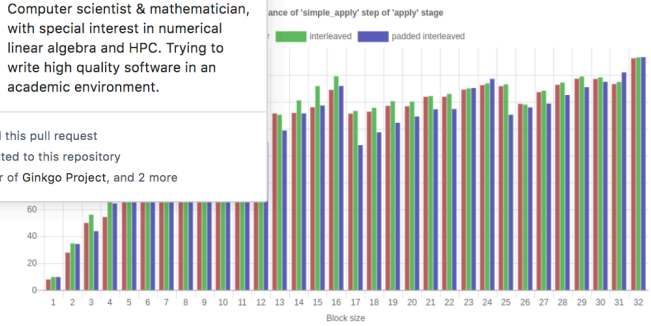
gflegar commented on Nov 18, 2018 — Author | Member

On vacation

Goran Flegar gflegar
Computer scientist & mathematician, with special interest in numerical linear algebra and HPC. Trying to write high quality software in an academic environment.

Opened this pull request
Committed to this repository
Member of Ginkgo Project, and 2 more

rleaved options were a bit slower, due to some strange
aved version, on the V100, interleaved storage (version 2)



I'll generate more details plots and send them around tomorrow.

gflegar merged commit 53f2c38 into `develop` on Nov 26, 2018 — View details
1 check passed

A good software patch has

- Motivation
- Technical explanation
- Performance evaluation
- Rigorous code review
- Code refactoring
- Multiple feedback loops

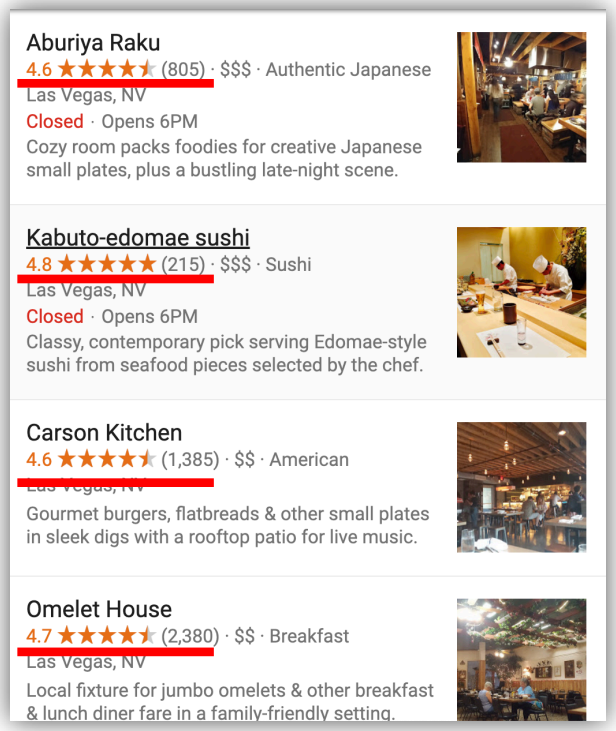
Ginkgo: https://github.com/ginkgo-project/ginkgo
Pull request #159 to add new features

Similar to the scientific journal contributions

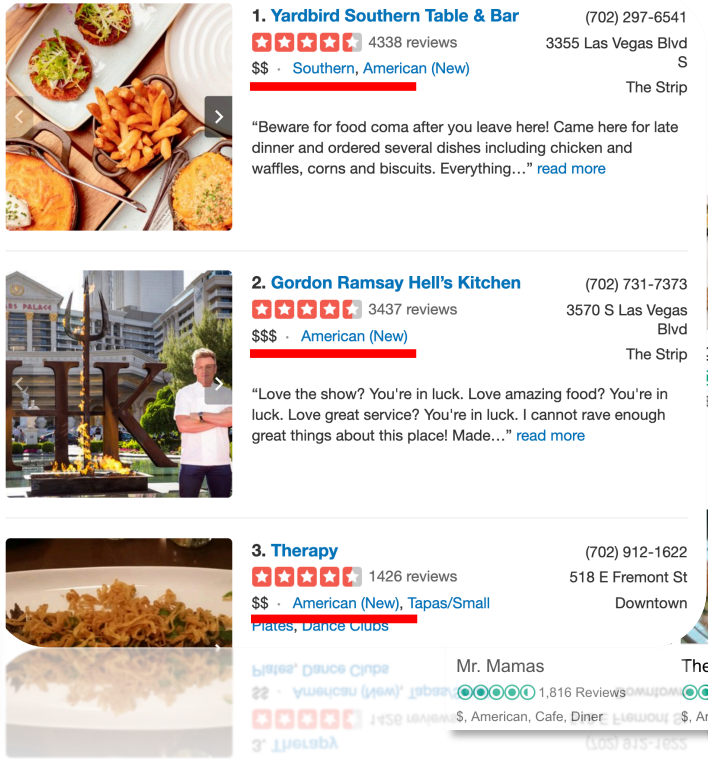
Interlude …

- I was finding a place to eat …

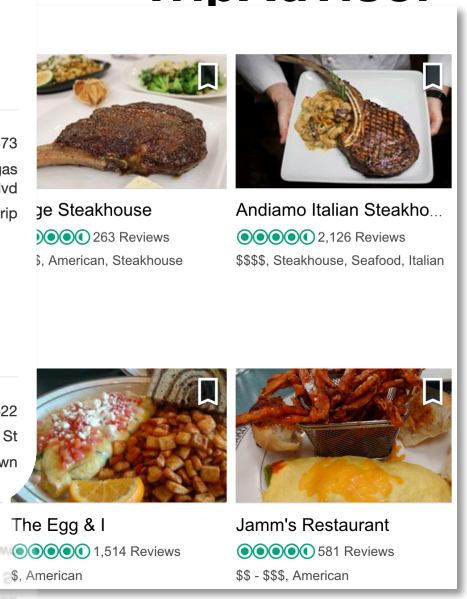
Google review



Yelp rating



TripAdvisor



"Too many places… Where do we go?"

"Let's go to the one with the highest star in the rating app!"

Advertise Your Project

No one knows you until you let others know …

- **Many users use your project because of stars** Star
  - Stars are the popularity and credibility of your project
  - Stars are an indicator of the number of potential users

Reply to: comment from linuxoidspb05/22/19 9:29:02 PM

> *Also very useful library for multithreading https://github.com/cpp-taskflow/cpp-taskflow*

Well, that is very popular (more than 1700 stars), unlike ...

anonymous (05/22/19 9:40:46 PM)

[Reply to this message] [Link]

linux.org.ur: https://www.linux.org.ru/news/development/15005663

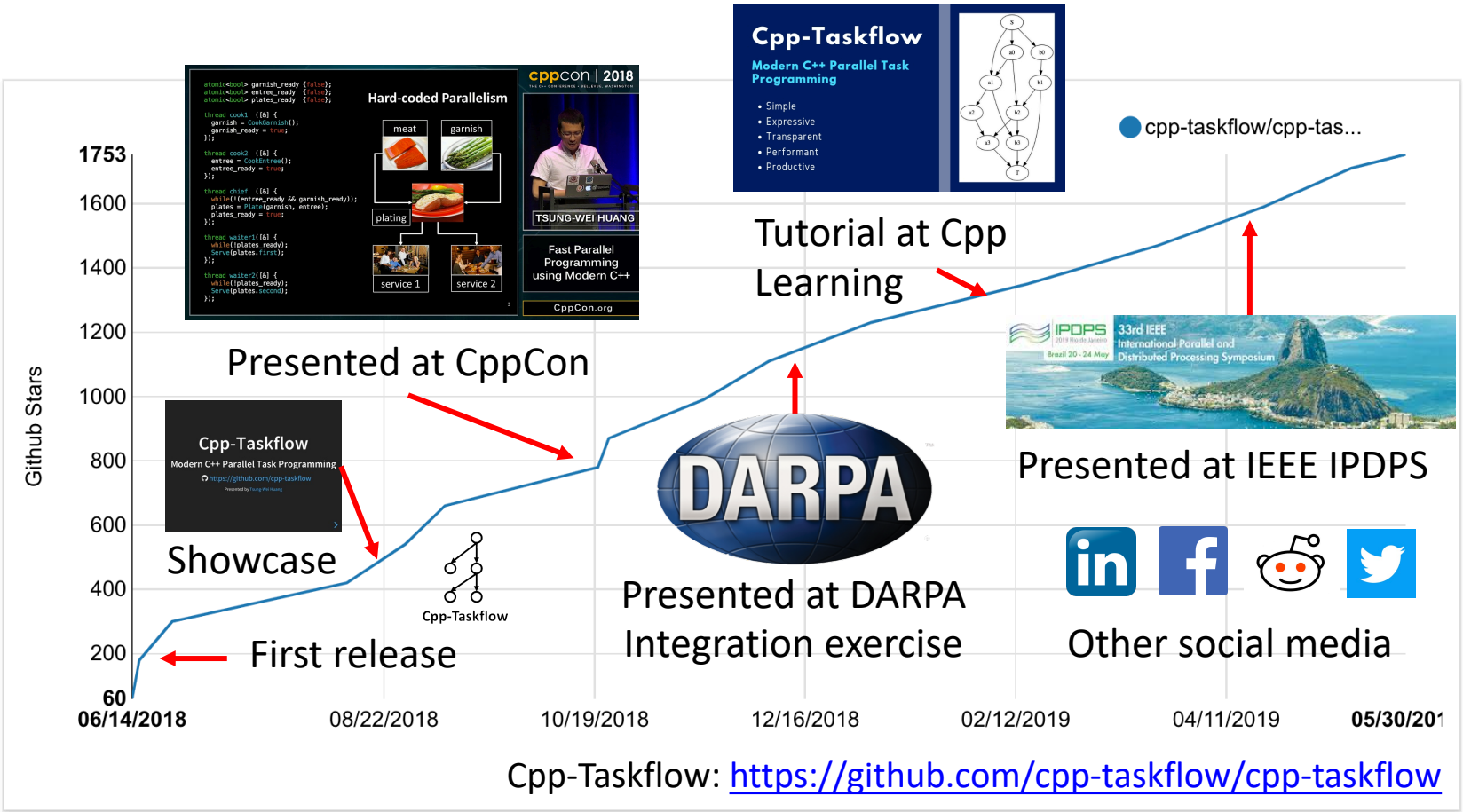
- **If you have a tasty cake, make it look tasty**
  - Add logo and badges to your README

Cpp-Taskflow

code quality A | build passing | build passing | c++ 17 | download latest | API documentation | awesome resources

- **Advertise the project multiple times for each release**

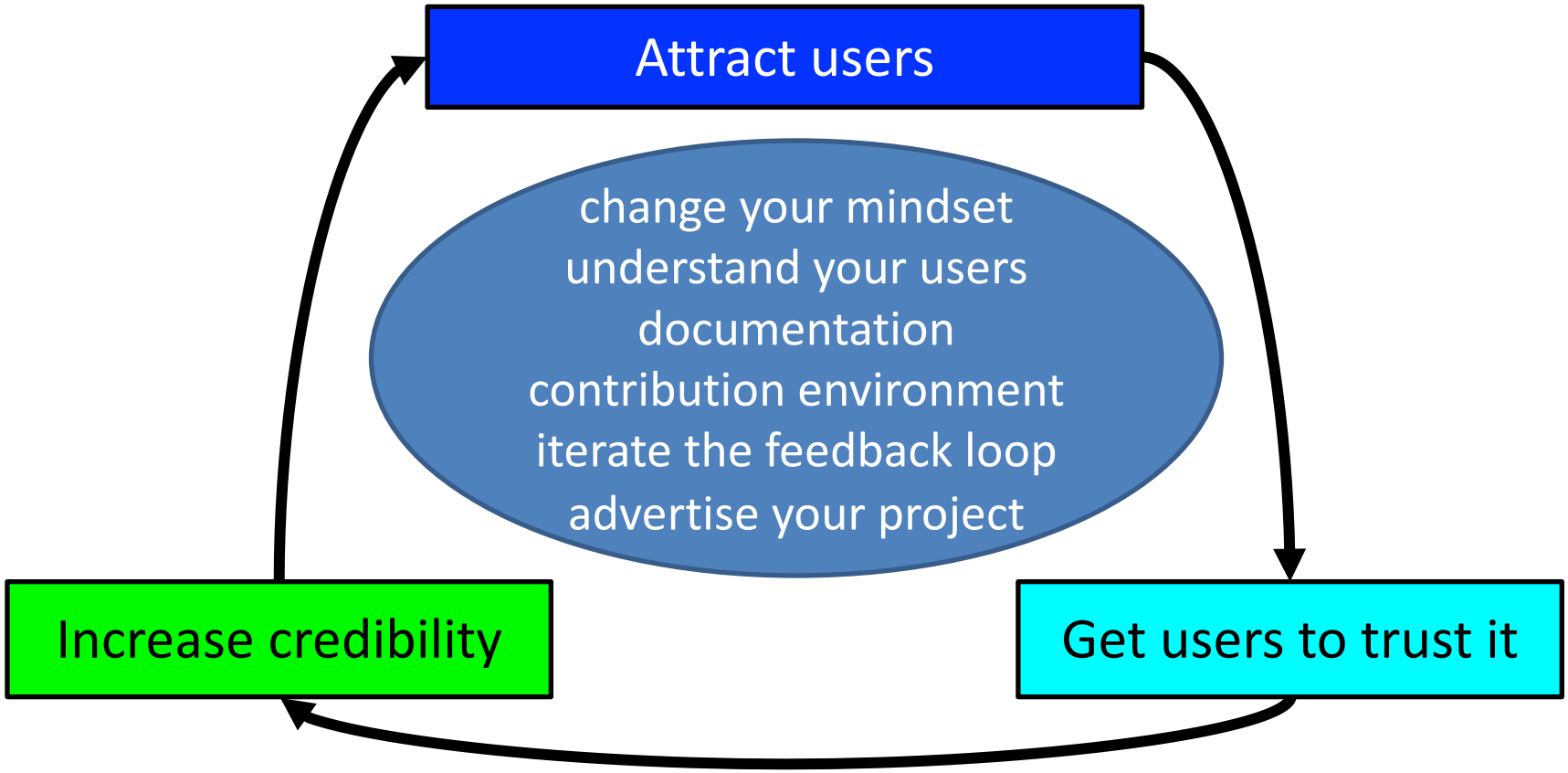
Cpp-Taskflow's Star History



Cpp-Taskflow: https://github.com/cpp-taskflow/cpp-taskflow

Advertising your project is important, but keep in mind it is your project content that makes people use it and like it

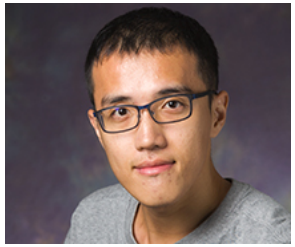
Conclusion: The Final Iron Circle



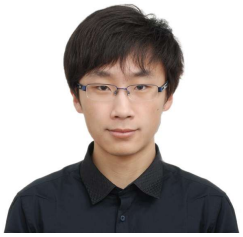
We should work together to change the current crediting system to reward software engineering & scientific software engineers

Thank You (and all our Users) ☺

T.-W. Huang



C.-X. Lin



G. Guo

M. Wong

A special thank goes to the DARPA IDEA program and their team for the support!

GitHub: https://github.com/tsung-wei-huang

Twitter: https://twitter.com/twh760812

Website: https://tsung-wei-huang.github.io/

Some Interesting Examples

- **End-users can be tricky …**

  "I can't understand why it is so difficult to compiler and install your tool on my platform. If you can't make it easy, how dare you open your project to waste people's time?" ***user A***

  "I am going to use tool xyz because yours really sucks." ***user B***

- **Developers might be trickier …**

  "If your job is to crash my tool, congratulations." ***developer A***

  "There are many other people using my project. It makes no difference to me to not include you." ***developer B***

- OpenTimer: A High-performance Timing Analysis Tool for Very Large Scale Integration (VLSI) Systems
- DtCraft: A General-purpose Distributed Programming Systems using Data-parallel Streams
- Firestorm: Fighting Game Engine with Asynchronous Resource Loaders (developed by ForgeMistress)
- Shiva: An extensible engine via an entity component system through scripts, DLLs, and header-only (C++)
- PID Framework: A Global Development Methodology Supported by a CMake API and Dedicated C++ Projects

More...

Contributors

Never forget to give your contributors credits!

Cpp-Taskflow is being actively developed and contributed by the following people:

- Tsung-Wei Huang created the Cpp-Taskflow project and implemented the core routines
- Chun-Xun Lin co-created the Cpp-Taskflow project and implemented the core routines
- Martin Wong supported the Cpp-Taskflow project through NSF and DARPA funding
- Andreas Olofsson supported the Cpp-Taskflow project through the DARPA IDEA project
- Nan Xiao fixed compilation error of unittest on the Arch platform
- Vladyslav fixed comment errors in README.md and examples
- vblanco20-1 fixed compilation error on Microsoft Visual Studio
- Glen Fraser created a standalone C++14-compatible threadpool for taskflow; various other fixes and examples
- Guannan Guo added different threadpool implementations to enhance the performance for taskflow
- Patrik Huber helped fixed typos in the documentation
- ForgeMistress provided API ideas about sharing the executor to avoid thread over-subscriptiong issues