

This Presentation Will Be Recorded

- ▶ By joining this Zoom web tutorial session, you automatically consent to the recording of all video, audio, and chat-room content.
- ▶ Furthermore, you grant permission to ACM and the OpenCilk organization to share the recordings, in full or in part, internally and with third parties.
- ▶ Please join without video and stay muted if you do not wish to be recorded.

Slides downloadable at

<http://opencilk.org/beta2/opencilk.spaa.2020.pdf>

Research and Teaching with OpenCilk

Dorothy Curtis
Alexandros–Stavros Iliopoulos
I–Ting Angelina Lee
Charles E. Leiserson
Tao B. Schardl
& many helpers

www.opencilk.org

contact@opencilk.org

*ACM Symposium on Parallelism in
Algorithms and Architectures*

July 14, 2020

Parallel-Computing Researchers



Eagles

Researchers employing large amounts of parallel computing in support of their scientific interests.



Owls

Researchers developing parallel-computing technology to address the future needs of Eagles.

Cilk is the only parallel-computing platform that effectively serves both Eagles and Owls.

Academic Impact of Cilk

Google scholar: 9150 citations

- ▶ Cilk runtime system [BJKLRZ96]: 2403 citations
- ▶ Cilk language [FLR98]: 1556 citations
- ▶ Cilk scheduler [BL99]: 1936 citations

Professional venues that have featured research papers that meaningfully rely on Cilk:

3PGCIC, ACM-SE, ACTAE, AIMS, ALENEX, ASPLOS, BIG DATA, CC, CF, CGO, COMPSAC, CSS, DAC, DCC, DFM, ICACT, ICCSE, ICDE, ICSS, ICPADS, ICPP, ICS, ICTAI, ICWC, IPDPS, ISCA, ISSAC, JACM, LLVM-HPC, MIPRO, OOPSLA/SPLASH, PACT, PASCO, PDP, PLDI, POPL, PPOPP, RTSS, SC, SIGCSE, SIGMETRICS, SIGOPS, SODA, SPAA, SoftCOM, TOCS, TOPC, TOPLAS, VLDB, VL/HCC, VPA, WOSC, and more.

Universities that have used Cilk in their courses:

Alabama, ANU, Binghamton, CMU, Cornell, Duke, Fudan, George Washington, Georgetown, Georgia Tech, Harvard, Indiana, Johannes Kepler, Knox College, Lehigh, Maryland, Michigan, MIT, NTU, NUS, Oregon, Otago, Oxford, Princeton, Purdue, Rice, Rochester, Rutgers, Stanford, Stonybrook, TU Wein, Tel Aviv, Texas, UC Berkeley, UCSB, UNC, Washington, WUSTL, Yale, and more.

A Brief History of Cilk Technology

1994–2006	Cilk project formed at MIT. Cilk offers simple C-based multithreaded programming combined with execution efficiency.
2006–2009	Cilk Arts spun out of MIT. Cilk++ provides support for C++, parallel loops, and reducer hyperobjects.
2009–2014	Intel Corporation acquires Cilk Arts. Cilk Plus offers Cilk++ and vector ops in ICC and GCC.
2014–2017	Due to attrition in Intel's Cilk team, the development of Cilk Plus at Intel stagnates.
2017	Intel announces it is dropping support for Cilk Plus, and GCC follows suit.
2019	NSF and Air Force fund development of OpenCilk.
2020	OpenCilk 1.0 beta released.

OpenCilk System Architecture

- ▶ **Compatibility** — Provide backward compatibility with Cilk Plus minus vector ops (i.e., Cilk++).
- ▶ **Open source** — Distribute under liberal open-source licenses.
- ▶ **Componentization** — Divide system into distinct software components with well-defined interfaces.
- ▶ **Integration** — As individual components are enhanced, ensure that they continue to interoperate with the entire platform.
- ▶ **Reliability** — Provide a suite of extensive tests and benchmarks to ensure that releases are stable, perform well, and are free of serious bugs.

OpenCilk Tutorial

- ▶ Our focus is on researchers and educators who already know something about Cilk.
- ▶ During the hands-on session, you can download and install (from binary) the OpenCilk 1.0 beta 2 release.
- ▶ To get a jump on things, download the latest release of OpenCilk here:
 - <https://github.com/OpenCilk/opencilk-project/releases>
- ▶ Technical-support assistants will hold your hand as you experiment with the release, including getting your legacy Cilk++ and Cilk Plus codes to work.
- ▶ You can also schedule additional technical support for some time in the future.

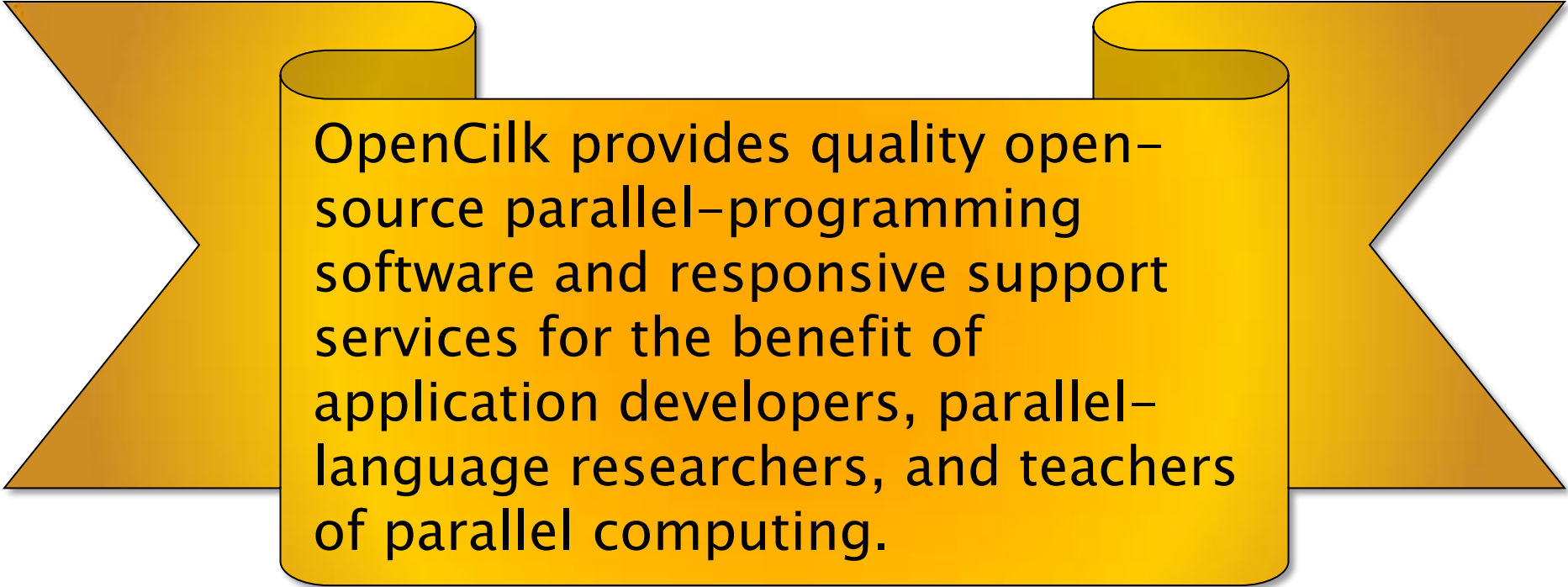
Outline

- ▶ Background of OpenCilk
- ▶ **The OpenCilk Organization**
- ▶ **Overview of OpenCilk 1.0 Beta 2**
- ▶ **Plans and Q&A**
- ▶ **[Break]**
- ▶ **Hand-Holding Hands-On Session**

Outline

- ▶ Background of OpenCilk
- ▶ **The OpenCilk Organization**
- ▶ **Overview of OpenCilk 1.0 Beta 2**
- ▶ **Plans and Q&A**
- ▶ **[Break]**
- ▶ **Hand-Holding Hands-On Session**

OpenCilk Mission



OpenCilk provides quality open-source parallel-programming software and responsive support services for the benefit of application developers, parallel-language researchers, and teachers of parallel computing.

OpenCilk operates under the auspices of MIT.

OpenCilk Development Team

Leadership

- ▶ Tao B. Schardl, MIT — Director, Chief Architect
- ▶ I-Ting Angelina Lee, WUSTL — Director, Runtime Architect
- ▶ John F. Carr, consultant — Senior Programmer
- ▶ Dorothy Curtis, MIT — Project Manager
- ▶ Charles E. Leiserson, MIT — Executive Director

Other contributors

- ▶ Alexandros-Stavros Iliopoulos, postdoc, MIT
- ▶ Tim Kaler, Ph.D. student, MIT
- ▶ Matthew Kilgore, Ph.D. student, MIT
- ▶ Billy Moses, Ph.D. student, MIT
- ▶ Kyle Singer, Ph.D. student, WUSTL
- ▶ Daniele Vettorel, Ph.D. student, MIT→Google
- ▶ Grace Yin, M.Eng. student, MIT→MIT

OpenCilk Advisory Board

- ▶ OpenCilk development activities are reviewed by the OpenCilk Advisory Board.
- ▶ Leadership
 - Vivek Sarkar, Georgia Tech — Chair
 - John Gilbert, UCSB — Co-Chair
 - Lawrence Rauchwerger, UIUC — Co-Chair
- ▶ Members
 - Umut Acar, Vikram Adve, David Bader, Pavan Balaji, Guy E. Blelloch, Aydın Buluç, David Bunde, Andrew Chien, Rezaul Chowdhury, Chen Ding, Alan Edelman, Jeremy T. Fineman, Matteo Frigo, Philip B. Gibbons, Pablo Halpern, Shahin Kamali, Bradley C. Kuszmaul, Will Leiserson, Marc Moreno Maza, Janice McMahon, John Mellor-Crummey, David Padua, Keshav Pingali, Nikos Pitsianis, Jan Prins, Nir Shavit, Julian Shun, Guy L. Steele Jr., Jim Sukha, Xiaobai Sun, Michael Bedford Taylor, Charles R. Tolle.

Outline

- ▶ Background of OpenCilk
- ▶ The OpenCilk Organization
- ▶ **Overview of OpenCilk 1.0 Beta 2**
- ▶ **Plans and Q&A**
- ▶ **[Break]**
- ▶ **Hand-Holding Hands-On Session**

OpenCilk 1.0 Beta 2

Open-source components

- ▶ Compiler (based on Tapir/LLVM)
- ▶ Runtime system (based on Cheetah)
- ▶ Compiler-based tools
 - Cilksan race detector
 - Cilkscale scalability analyzer
- ▶ Regression tests and benchmark suite
- ▶ Intel's Cilk Plus reducer library

Task-parallel features

- ▶ Full support for exceptions
- ▶ No support for vector extensions (intended)
- ▶ No support for pedigrees (coming)
- ▶ Ability to spawn statement blocks (new)

Current Target Platforms

- ▶ Unix/Linux x86-64
- ▶ Tested on the following releases
 - Ubuntu 18.04
 - FreeBSD 12.1
 - Fedora 30
 - MacOSX 10.15
- ▶ May work on other distributions/releases
 - Ask us!

The OpenCilk Compiler

The OpenCilk compiler is based on the award-winning Tapir/LLVM compiler [SML17].

- ▶ Compile Cilk programs using `clang` and the `-fopencilk` flag.

```
$ clang fib.c -o fib -O3 -fopencilk
$ ./fib 35
```

- ▶ Tapir/LLVM optimizes Cilk programs more effectively than GCC and ICC.
- ▶ The OpenCilk compiler contains bug fixes and performance improvements over the original Tapir/LLVM compiler.

Summary of Compiler Features

Beta 2 capabilities

- ▶ Based on Clang and LLVM 9.
- ▶ Supports the Cilk keywords `cilk_spawn`, `cilk_sync`, and `cilk_for` in C/C++ programs.
- ▶ Also supports `cilk_spawn` of statements other than function calls:
 - E.g., `cilk_spawn { x += y; } .`
- ▶ Works with standard Clang flags, including optimization and debugging flags.

Status

- ▶ Tested on a wide variety of Cilk applications, including Cilk-5 applications, PBBS, and Ligra.
- ▶ For C code, earlier versions of the compiler were battle-tested by hundreds of students.

The OpenCilk Runtime System

Beta 2 capabilities

- ▶ Provides a simple, easy-to-extend, and high-performing work-stealing runtime system.
- ▶ Based on Cheetah (WUSTL).
- ▶ The scheduler supports `cilk_spawn`, `cilk_sync`, `cilk_for`, exceptions, and reducer hyperobjects.

Status

- ▶ Source lines of code < 5,000, including comments and core header files.
 - Compare to Cilk Plus's 22,000+.
- ▶ Performance currently lags slightly behind Cilk Plus if the application has limited parallelism.

Reducer Hyperobjects

Beta 2 capability

- Provides the same linguistic interface and functionality as Cilk Plus.

Status

- Uses Cilk Plus's reducer library.
- Runtime contains a more-efficient reducer data structure, but performance can still be improved.
- Hard limit on the number of active reducers.

Cilksan Race Detector

Beta 2 capabilities

- ▶ Open-source replacement for Intel's closed-source Cilkscreen race detector.
- ▶ Basic determinacy-race detection on memory accesses.
- ▶ Uses compiler instrumentation to implement a serial "SP-bags" algorithm.

Status

- ▶ Tested on ~20 Cilk applications.
- ▶ No special support for locks and reducers.
- ▶ Earlier versions were battle-tested by hundreds of students doing Cilk programming.

Cilksan Example

- ▶ Find logically parallel accesses to the same location (at least 1 write).
- ▶ Analysis cost proportional to serial execution:
 - $\sim 7\times$ overhead on this example.

```
[...]
b = (char*) alloca((j+1) * sizeof(char));
memcpy(b, a, j * sizeof(char));
for (int i = 0; i < n; i++) {
    b[j] = i; /* <-- racy write! */
    if (ok(j+1,b))
        cnt[i] = cilk_spawn nqueens(n,j+1,b);
}
[...]
```

nqueens.c

```
$ ./nqueens 12
Running Cilksan race detector
Running ./nqueens with n = 12.
Race detected at address 7f7db6c0f2e6
*   Read 43ef18 nqueens ./nqueens.c:87:3
|   `--to variable a (declared at nqueens.c:50)
+   Call 43f73b nqueens ./nqueens.c:91:29
+   Spawn 43efd7 nqueens ./nqueens.c:91:29
|*  Write 43efa9 nqueens ./nqueens.c:89:10
||   `--to variable b (declared at ./nqueens.c:53)
\| Common calling context
+   Call 43f73b nqueens ./nqueens.c:91:29
+   Spawn 43efd7 nqueens ./nqueens.c:91:29
[...]
+   Call 43f42b main ./nqueens.c:125:9
Allocation context
Stack object b (declared at ./nqueens.c:53)
Alloc 43eef8 in nqueens ./nqueens.c:86:16
Call 43f73b nqueens ./nqueens.c:91:29
Spawn 43efd7 nqueens ./nqueens.c:91:29
[...]
Call 43f42b main ./nqueens.c:125:9

2.544000
Total number of solutions : 14200

Race detector detected total of 1 races.
Race detector suppressed 3479367 duplicate error
messages
```

Cilkscale Scalability Analyzer

Beta 2 capabilities

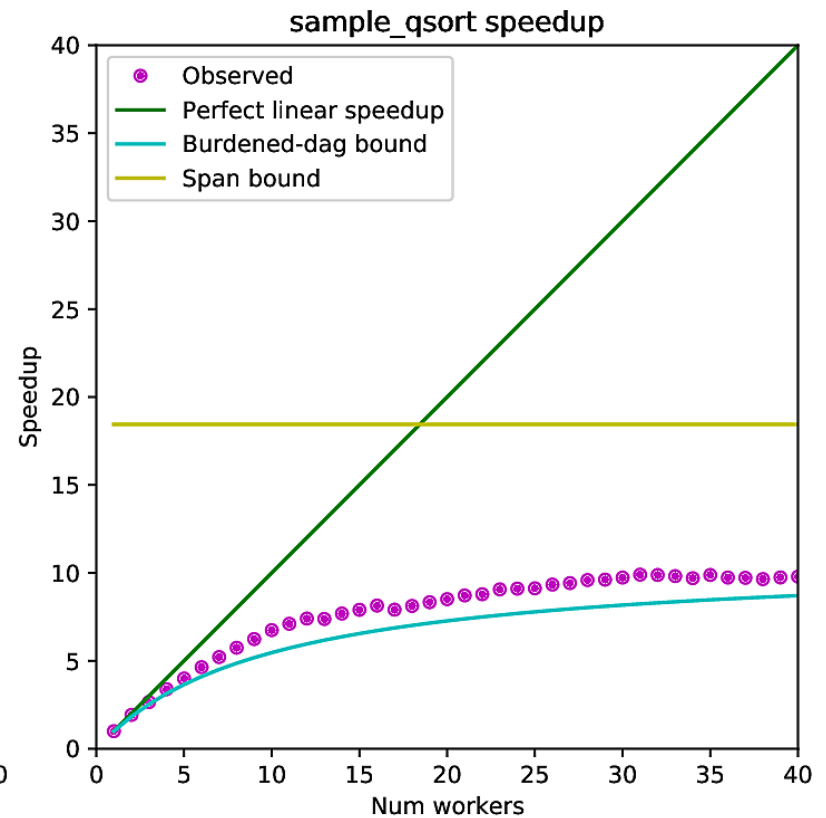
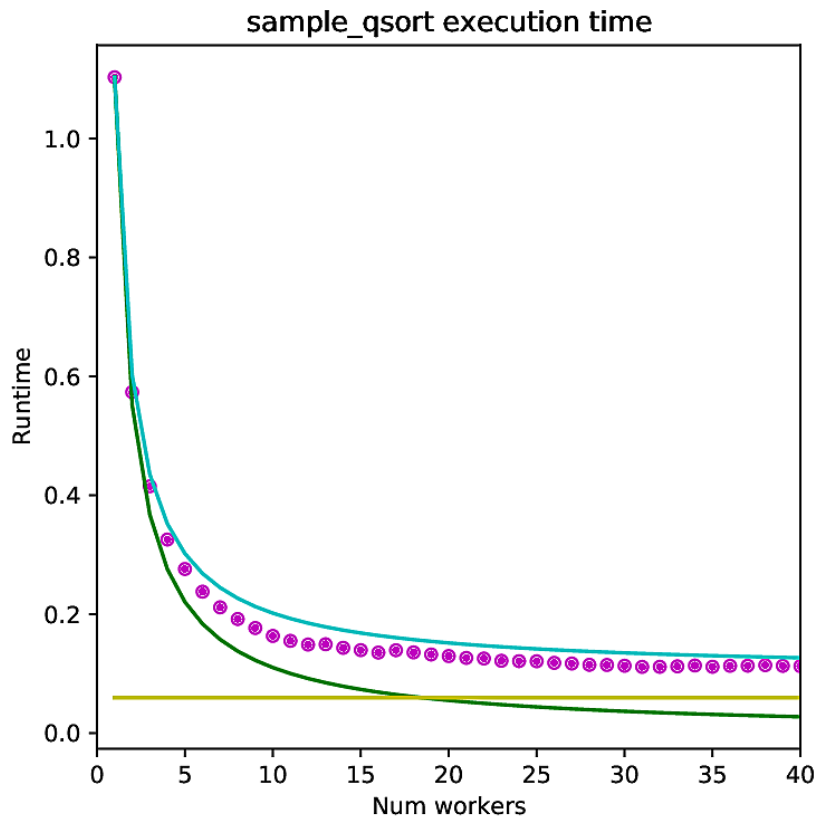
- ▶ Open-source replacement for Intel's closed-source Cilkview scalability analyzer.
- ▶ Analyze whole-program or user-specified region.
- ▶ Automated benchmarking, visualizer, CSV output.

Status

- ▶ Tested on ~5 Cilk applications.
- ▶ Earlier versions were battle-tested by hundreds of students doing Cilk programming.

Cilkscale Visualizer Output

Example: Parallel Quicksort



Outline

- ▶ Background of OpenCilk
- ▶ The OpenCilk Organization
- ▶ Overview of OpenCilk 1.0 Beta 2
- ▶ **Plans and Q&A**
- ▶ **[Break]**
- ▶ **Hand-Holding Hands-On Session**

OpenCilk Releases

- ▶ OpenCilk 1.0 beta 1 — spring 2020
- ▶ OpenCilk 1.0 beta 2 — summer 2020
- ▶ OpenCilk 1.0 beta 3 — late summer 2020
- ▶ OpenCilk 1.0 — late fall 2020
- ▶ OpenCilk 2.0 — ?

OpenCilk 1.0 and Beyond

- ▶ Performance engineering
- ▶ Pedigrees and deterministic pseudorandom-number generator
- ▶ Enhanced componentization, including pluggable scheduler
- ▶ Compiler-based instrumentation framework
- ▶ Programmer start-up of runtime system
- ▶ Multiple cilk
- ▶ Improved reducer syntax
- ▶ Enhanced and new productivity tools
- ▶ OpenCilk user groups (Eagles and Owls)
- ▶ ⟨Your idea here⟩

Q&A

Outline

- ▶ Background of OpenCilk
- ▶ The OpenCilk Organization
- ▶ Overview of OpenCilk 1.0 Beta 2
- ▶ Plans and Q&A
- ▶ **[Break]**
- ▶ **Hand-Holding Hands-On Session**

Outline

- ▶ Background of OpenCilk
- ▶ The OpenCilk Organization
- ▶ Overview of OpenCilk 1.0 Beta 2
- ▶ Plans and Q&A
- ▶ [Break]
- ▶ **Hand-Holding Hands-On Session**

Demo: Download OpenCilk

Download the latest OpenCilk release:

- ▶ <https://github.com/OpenCilk/opencilk-project/releases>

Installation

- ▶ Linux binaries can be installed using `OpenCilk-9.0.1-Linux.sh`.
- ▶ Binaries for MacOSX 10.14 or newer can be installed using `OpenCilk-9.0.1-Darwin.sh`.
- ▶ Instructions to build OpenCilk from source:
 - <https://github.com/OpenCilk/infrastructure>

Download the tutorial code examples

- ▶ <https://github.com/OpenCilk/tutorial>

Demo: Compile and Run

To compile a program with OpenCilk, pass the `-fopencilk` flag to Clang:

```
$ clang fib.c -o fib -O3 -fopencilk  
$ ./fib 35
```

Demo: Using Cilksan

Add the `-fsanitize=cilk` compiler flag to enable the Cilksan determinacy-race detector:

```
$ clang nqueens.c -o nqueens -fopencilk -fsanitize=cilk -Og -g  
$ ./nqueens 12
```

We recommend the `-Og -g` flags for debugging.

Note: On **MacOSX**, the compiler compiles `nqueens.c` using builtins that Cilksan does not currently recognize. You can work around this behavior using the flag `-D_FORTIFY_SOURCE=0`:

```
$ clang nqueens.c -o nqueens -fopencilk -fsanitize=cilk -Og -g \  
> -D_FORTIFY_SOURCE=0
```


Demo: Using Cilkscale

Add the `-fcilktool=cilkscale` compiler flag to measure work and span using Cilkscale:

```
$ clang qsort.c -o qsort -fopencilk -fcilktool=cilkscale -O3  
$ ./qsort 10000000
```

Demo: Analyze a Region

Annotate the code using the Cilkscale API:

```
#include <cilk/cilkscale.h>

int main(int argc, char **argv) {
    ...
    wsp_t start, end;
    start = wsp_getworkspan();
    sample_qsort(a, a + n);
    end = wsp_getworkspan();
    ...
    wsp_dump(wsp_sub(end, start), "sample_qsort");
    ...
}
```

qsort.c

Recompile and rerun using Cilkscale:

```
$ clang qsort.c -o qsort -fopencilk -fcilktool=cilkscale -O3
$ ./qsort 10000000
```

Demo: Download the Cilkscale Visualizer

Download OpenCilk productivity-tools repository:

- ▶ <https://github.com/OpenCilk/productivity-tools>

```
$ git clone https://github.com/OpenCilk/productivity-tools.git
```

Demo: Cilkscale Visualizer

Compile the program twice,

- ▶ once with `-fcilktool=cilkscale`, and
- ▶ once with `-fcilktool=cilkscale-benchmark`:

```
$ clang qsort.c -o qsort -fopencilk -fcilktool=cilkscale -O3  
$ clang qsort.c -o qsort-bench -fopencilk -O3 \  
> -fcilktool=cilkscale-benchmark
```

Run the program with the visualizer:

```
$ cd productivity-tools/Cilkscale_vis  
$ python3 cilkscale.py -c ../../qsort -b ../../qsort-bench \  
> --args 10000000 -rplot 0,1
```

Open `plot.pdf` to view the performance plot.

Thank
you

www.opencilk.org
contact@opencilk.org

Support Acknowledgments

- ▶ **National Science Foundation:** OpenCilk development is supported in part by the National Science Foundation under Grant No. CNS-1925609. Any opinions, findings, and conclusions or recommendations expressed in this tutorial are those of the presenters and do not necessarily reflect the views of the National Science Foundation.
- ▶ **United States Air Force Research Laboratory:** OpenCilk development is supported in part by the United States Air Force Research Laboratory and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this tutorial are those of the presenters and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute content for Government purposes notwithstanding any copyright notation herein.