TEXAS TECH UNIVERSITY

# CS4379: Parallel and Concurrent Programming
# CS5379: Parallel Processing

## Lecture 14

**Dr. Yong Chen**

**Associate Professor**

**Computer Science Department**

**Texas Tech University**

# Course Info

- **Lecture Time**: TR, 12:30-1:50
- **Lecture Location**: ECE 217
- **Sessions**: CS4379-001, CS4379-002, CS5379-001, CS5379-D01
- **Instructor:** Yong Chen, Ph.D., Associate Professor
- **Email:** yong.chen@ttu.edu
- **Phone:** 806-834-0284
- **Office**: Engineering Center 315
- **Office Hours**: 2-4 p.m. on Wed., or by appointment
- **TA:** Mr. Ghazanfar Ali, Ghazanfar.Ali@ttu.edu
- **TA Office hours:** Tue. and Fri., 2-3 p.m., or by appointment
- **TA Office:** EC 201 A
- **More info:**
  - http://www.myweb.ttu.edu/yonchen
  - http://discl.cs.ttu.edu; http://cac.ttu.edu/; http://nsfcac.org

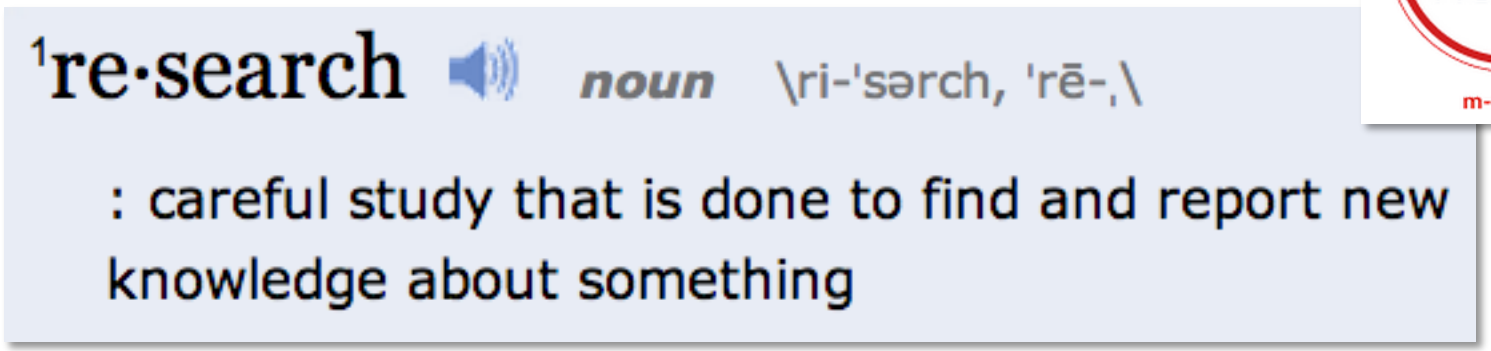
# Announcements

- Guest lectures by Mr. Wei Zhang on 3/5 and 3/10
- No class on 3/12, Thursday, due to computer science department one-day retreat activities

# Outline

- Questions?
- Course research project - conducting research and development and becoming a lifelong learner (cont.)
- Overview of programming models and thread basics
- The POSIX Threads

# What is Research?




- To discover unknown and create new knowledge
- Research is not "*search*", innovation is not (only) "*integration*"

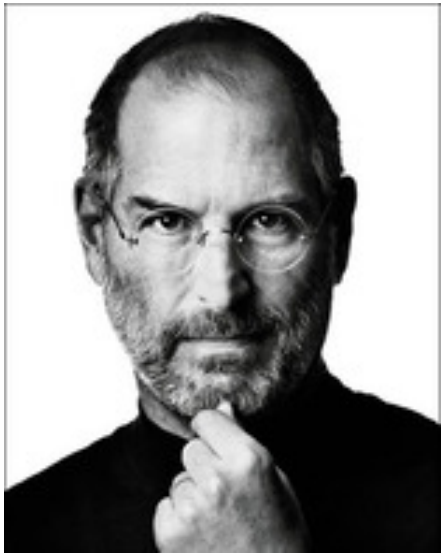
# Why Research?

Knowledge is Power.

- Sir Francis Bacon

"We're here to put a dent in the universe. Otherwise why else even be here?"

- Steve Jobs

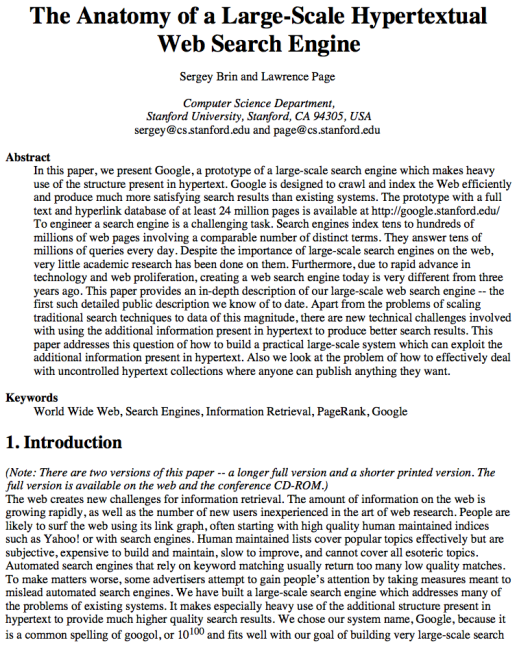


# Analogies

- Reading a book v.s. writing a book
- Reading a poem v.s. writing a poem (literally for arts and humanities research)
- Listening to a song v.s. composing a song including lyrics
- Watching a movie v.s. directing a movie
- Discover unknown and Create new

# Why Research? (cont.)

- Understand the nature, e.g. discovered the electricity, drugs, etc.
- Create new technologies, e.g. the internet, web search engine, etc.
- Can fundamentally change human's life, e.g. finding possible cure for cancer, mobile devices and cloud computing, HPC enabled simulations



**The Anatomy of a Large-Scale Hypertextual Web Search Engine**

Sergey Brin and Lawrence Page

*Computer Science Department,*
*Stanford University, Stanford, CA 94305, USA*
sergey@cs.stanford.edu and page@cs.stanford.edu

**Abstract**

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/ To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale web search engine -- the first such detailed public description we know of to date. Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.

**Keywords**

World Wide Web, Search Engines, Information Retrieval, PageRank, Google

**1. Introduction**

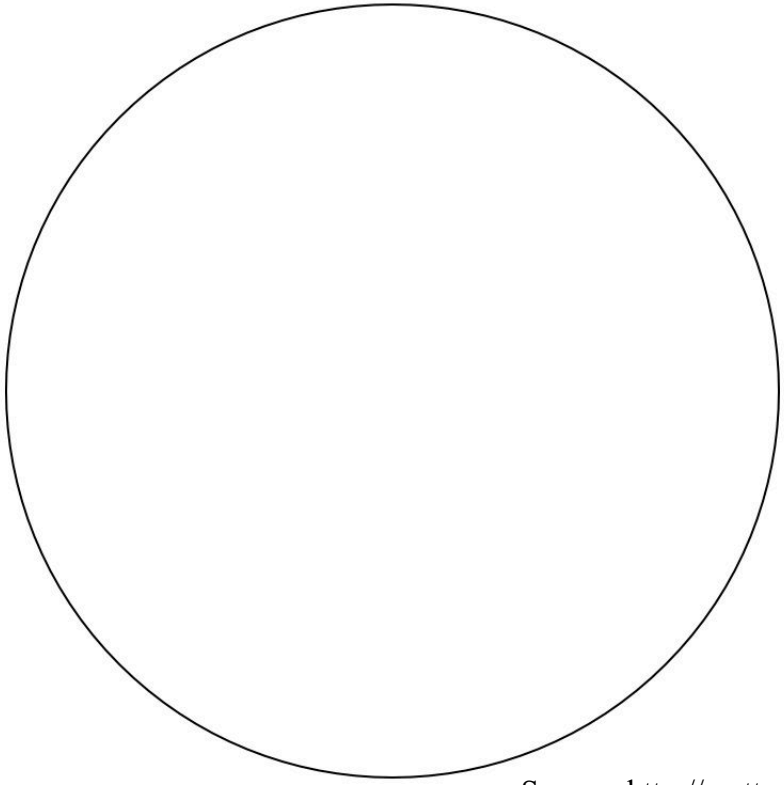
*(Note: There are two versions of this paper -- a longer full version and a shorter printed version. The full version is available on the web and the conference CD-ROM.)*
The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained indices such as Yahoo! or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics. Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead automated search engines. We have built a large-scale search engine which addresses many of the problems of existing systems. It makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results. We chose our system name, Google, because it is a common spelling of googol, or $10^{100}$ and fits well with our goal of building very large-scale search

Innovation has no limits. The only limit is your imagination. It's time for you to begin thinking out of the box. If you are involved in a growing industry, think of ways to become more efficient; more customer friendly; and easier to do business with. If you are involved in a shrinking industry – get out of it quick and change before you become obsolete; out of work; or out of business. And remember that procrastination is not an option here. Start innovating now!

There is no shortcut to excellence. You will have to make the commitment to make excellence your priority. Use your talents, abilities, and skills in the best way possible and get ahead of others by giving that little extra. Live by a higher standard and pay attention to the details that really do make the difference. Excellence is not difficult - simply decide right now to give it your best shot - and you will be amazed with what life gives you back.

# An Illustrated Guide (Matt Might)

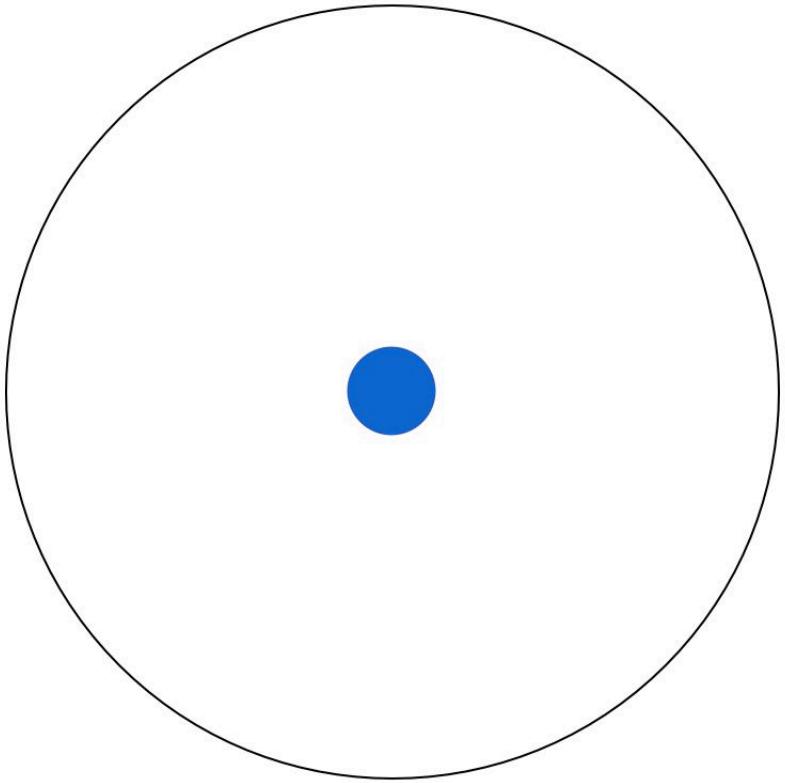
- Imagine a circle that contains all of human knowledge:



Source: http://matt.might.net/articles/phd-school-in-pictures/

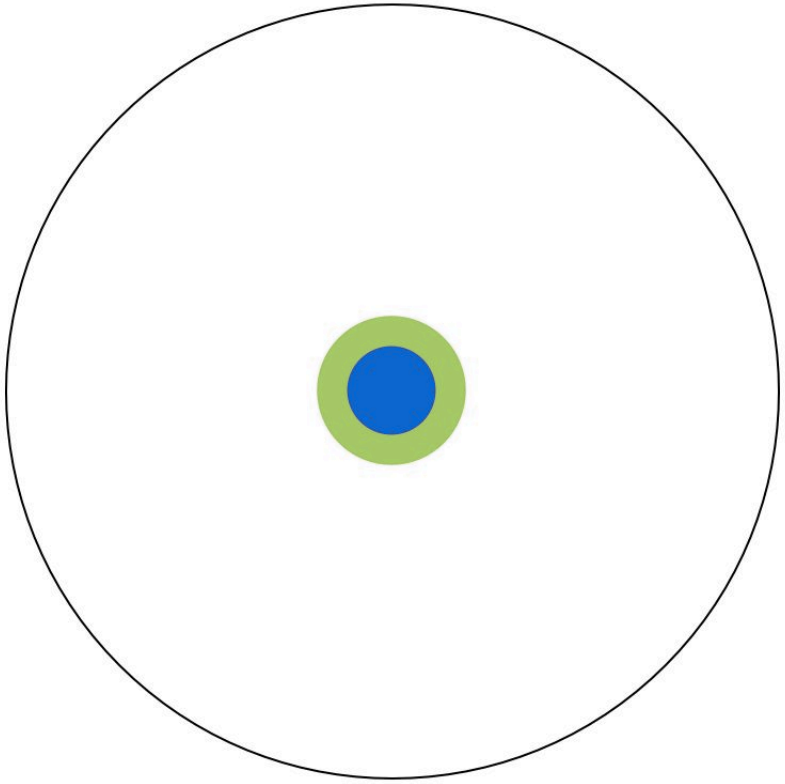
# An Illustrated Guide (Matt Might)

- By the time you finish elementary school, you know a little:



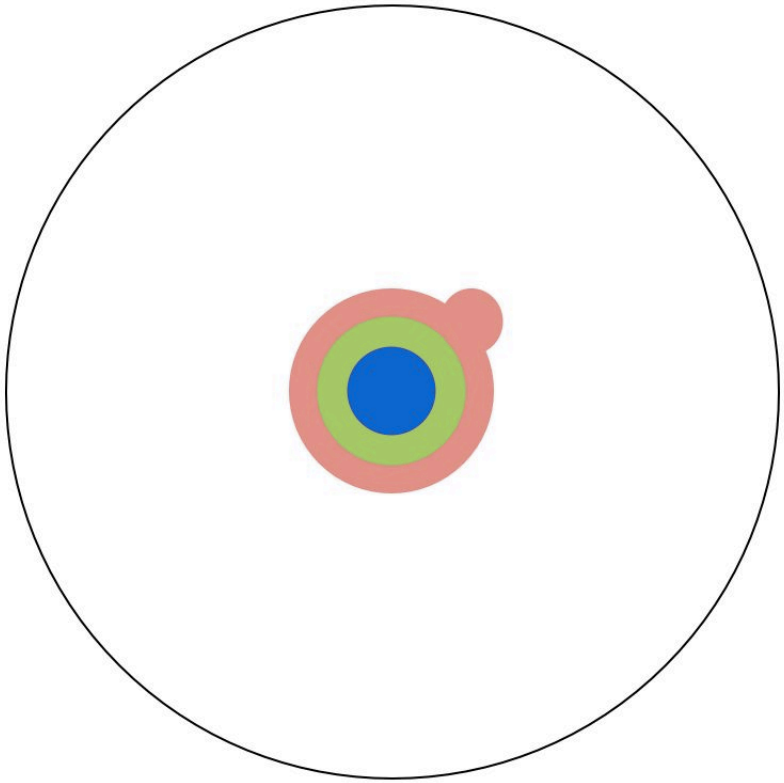
# An Illustrated Guide (Matt Might)

- By the time you finish high school, you know a bit more:



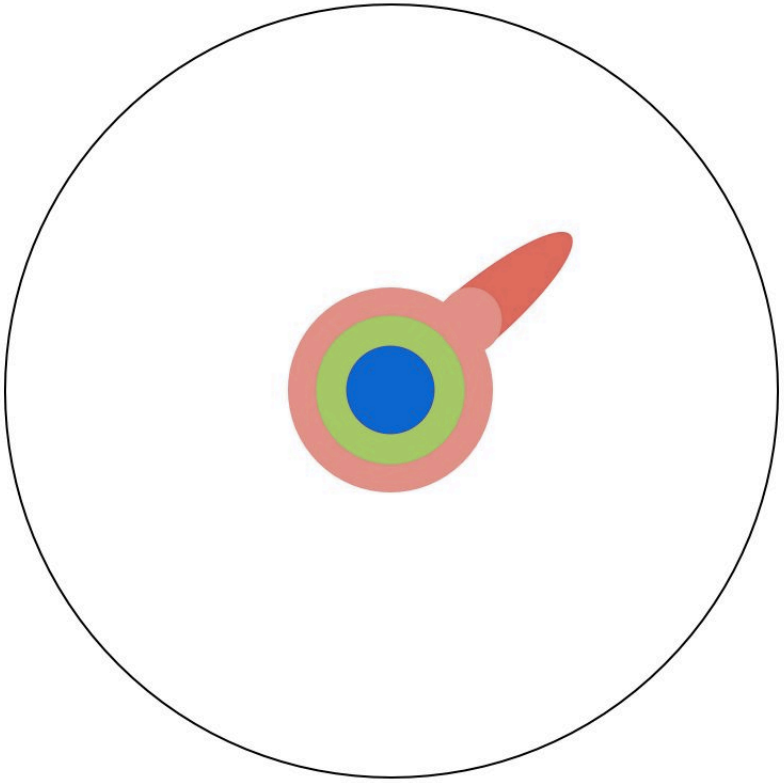
# An Illustrated Guide (Matt Might)

- With a bachelor's degree, you gain a specialty:



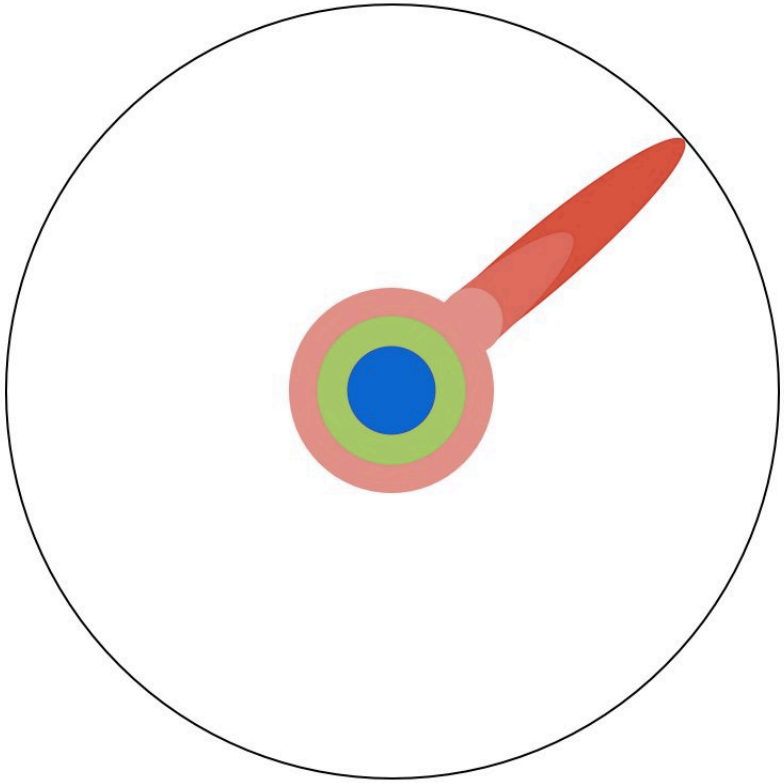
# An Illustrated Guide (Matt Might)

- A master's degree deepens that specialty:



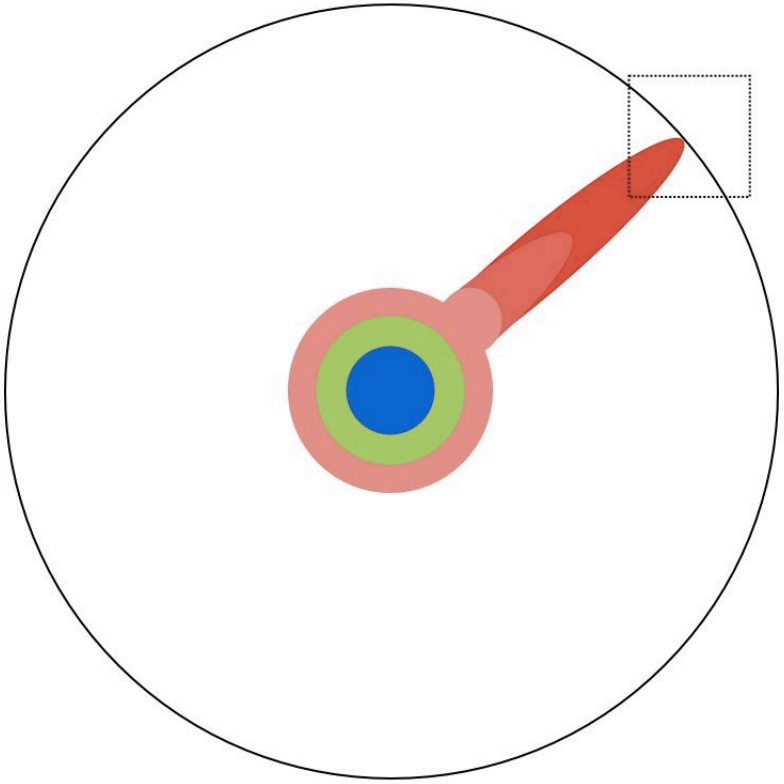
# An Illustrated Guide (Matt Might)

- Reading research papers takes you to the edge of human knowledge:



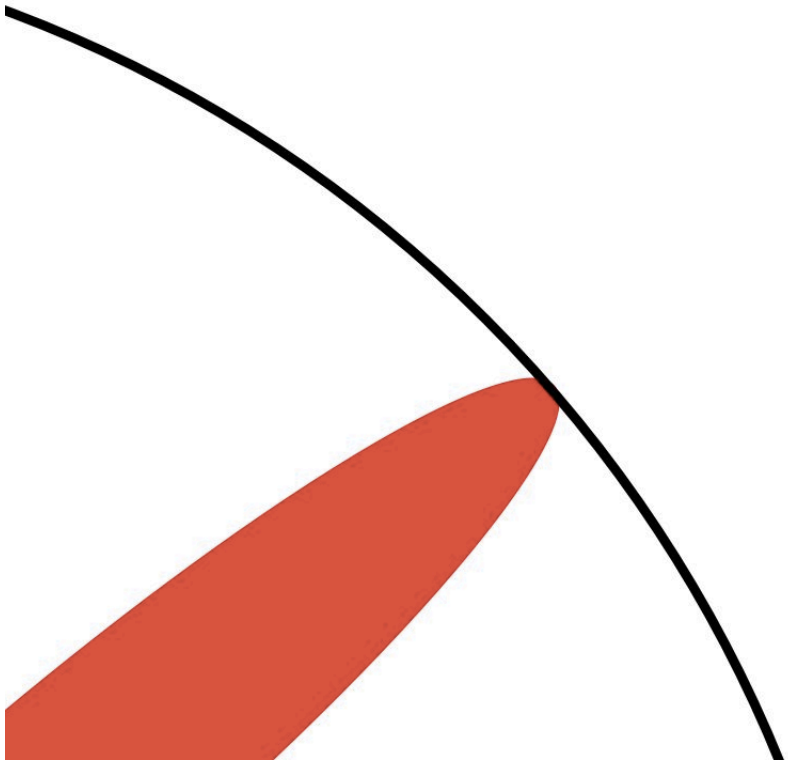
# An Illustrated Guide (Matt Might)

- Once you're at the boundary, you focus:



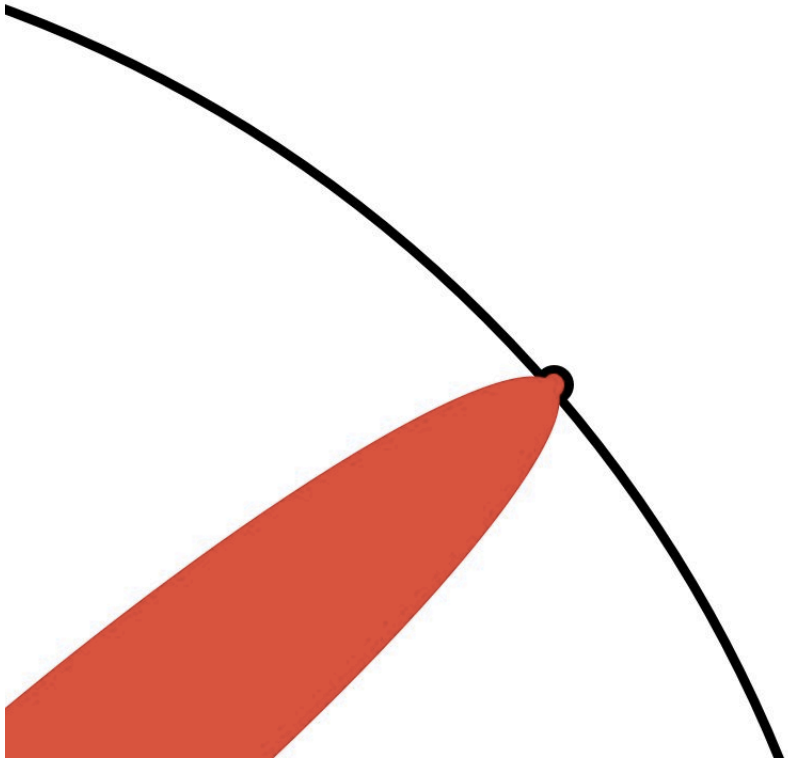
# An Illustrated Guide (Matt Might)

- You push at the boundary for a few years:



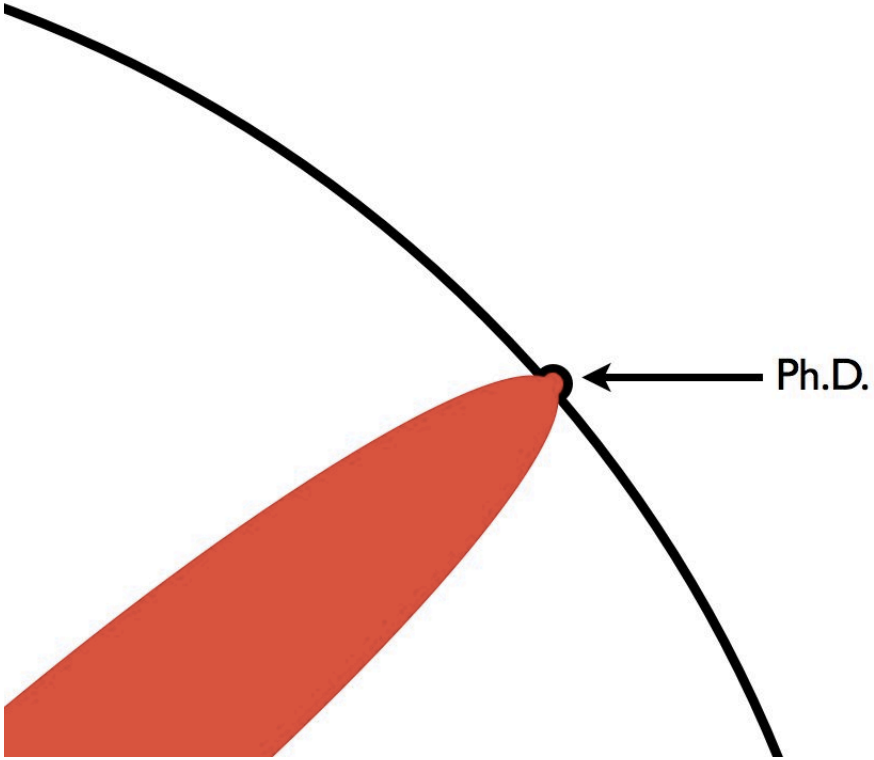
# An Illustrated Guide (Matt Might)

- Until one day, the boundary gives way:



# An Illustrated Guide (Matt Might)

- And, that dent you've made is called a Ph.D.:

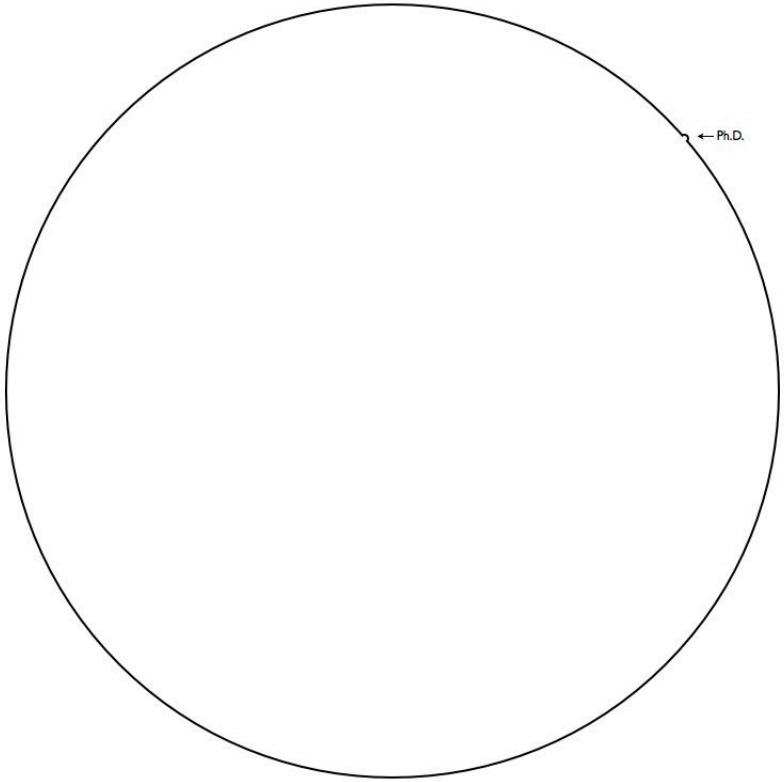


# An Illustrated Guide (Matt Might)

- Of course, the world looks different to you now:

# An Illustrated Guide (Matt Might)

- So, don't forget the bigger picture:



# Why Should I Pursue Research?

- You have enthusiasm in discovering unknown and enjoy the excitement
  - E.g. your work is shared and used by others, written in the textbook
  - You get excited if your findings get published, cited by others, talked about by others
  - Earned awards/grants, earned VC for commercialization and technology transfer, startup business
  - Gets to travel the world to present your findings

# Why Should I Pursue Research? (cont.)

- For career profession in academia or industry/government research labs ("researcher")
  - Nearly a must
  - Academia freedom
  - Job stability (potentially more values when growing older)
  - Own credit for all what you do (compared to the codes/products belong to the company as an engineer/developer)
  - Consulting, startup opportunities, relatively easy to move to industry v.s. the other way around

# Why Should I Pursue Research?

- For career profession in industry ("engineer/developer")
  - Training for problem identification and problem solving skills, proposal and presentation skills (developing convincing arguments), create new solutions/products, convince and compete for resources
  - Stand out from peers, grow to become a group lead, technical fellow, CEO/CTO
  - Rather than programmer/coder: given specification, turn into codes
  - Critical thinking, vision development, time management, execution training, networking (conference, presentation)

# A Typical Workflow of Conducting a Research Project

1. Identify a problem/build a hypothesis, review approaches
2. Propose a new approach; define, design, refine
3. Evaluate your approach[1]

   *(Production implementation), reference implementation, proof-of-concept prototype, emulation/demo, simulation/demo, theoretical proof/analysis*
4. Iterations of step 2 and 3
5. Write research papers for conference/journal publications
6. Present your research for dissemination and impact

[1] *https://discl.cs.ttu.edu/doku.php?id=evaluation_methodologies*

# Where do ideas come from?

- Problem driven
  - E.g. performance bottleneck (where does the bottleneck exist in a system?); enhance some aspect: efficiency, accuracy, coverage, reliability, availability, scalability, etc.; reduce some aspect: time/space complexity, cost (hardware/power/human efforts), etc.
- Application driven
  - Develop for new needs, or create new demands, new features, e.g. new needs for data-intensive/big data problems
  - "data-centric" solutions, paradigm change
- Technology driven
  - Embrace technology changes and leverage new technologies, evaluate and utilize technologies, stay keen to technologies
  - E.g. HDDs -> SSDs, DRAM -> 3-d stacked memory, exa-scale computing

## More Notes

- Use examples to motivate the research/idea
- Use diagrams/visual aid to illustrate the problem/idea/solution (simple and effective)
- Solid proof-of-concept evaluation: justifications
  - Theoretical proof, modeling and analysis, simulation, emulation, proof-of-concept prototype, reference implementation
- Comprehensive literature coverage and comparison
- Literature is your best sample
- Well execute!
  - **"Recipes" v.s. "Meals"**

# More Notes (cont.)

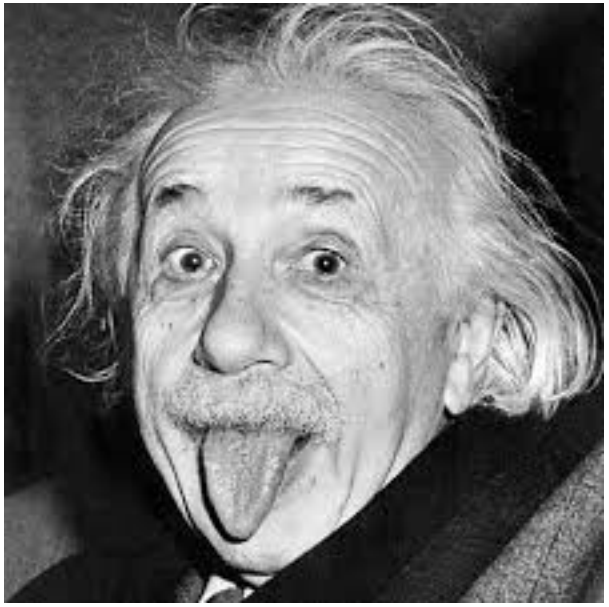
- How to read papers and write reviews
  - How to Read a Paper: http://ccr.sigcomm.org/online/files/p83-keshavA.pdf
  - How to Read a Research Paper: http://www.eecs.harvard.edu/~michaelm/postscripts/ReadPaper.pdf
  - The Task of the Referee: http://pages.cs.wisc.edu/~markhill/the_task_of_the_referee.pdf
  - Writing Reviews for Systems Conferences: http://people.inf.ethz.ch/troscoe/pubs/review-writing.pdf

## Summary

- Research is to discover unknown and to create new knowledge
  - Significant impact (and a challenging task too)
- Knowledge need to be conveyed and passed along in written form and writing research papers is critical for dissemination
  - Arguably a fundamental skill for students, particularly Phd and Masters' thesis students, and professional researchers
- "Execution" can be the most important, well execute and deliver!

"If we knew what it was we were doing, it would not be called research, would it?"

- Albert Einstein



# Keep pushing.

# Outline

- Questions?
- Course research project - conducting research and development and becoming a lifelong learner (cont.)
- Overview of programming models and thread basics
- The POSIX Threads
  - Creation and termination
  - Pthreads Examples

# Overview of Programming Models

- Parallel programming models (any) focus on providing support for expressing (via primitives/API):

- Concurrency
  - Create multiple processes/threads

- Synchronization
  - Coordinate these processes/threads to ensure correct results

# Overview of Programming Models

- Programming shared-address-space architectures
  - Communications implicitly
  - Can have process based, or thread based, or directive based
  - Process based models assume that all data associated with a process is private, by default, unless otherwise specified
    - E.g. Global Arrays, Unified Parallel C (UPC), OpenSHMEM, or GAS (Global Address Space)/PGAS (Partitioned Global Address Space)
  - Thread based models assume that all memory is global, faster manipulation, preferred model
  - Directive based models extend the threaded model by facilitating creation and synchronization of threads using directives

# Overview of Programming Models

- A *thread* is a single stream of control in the flow of a program.

- A program like:

```
for (row = 0; row < n; row++)
  for (column = 0; column < n; column++)
        c[row][column] =
        dot_product( get_row(a, row),
                     get_col(b, col));
```

can be transformed to:

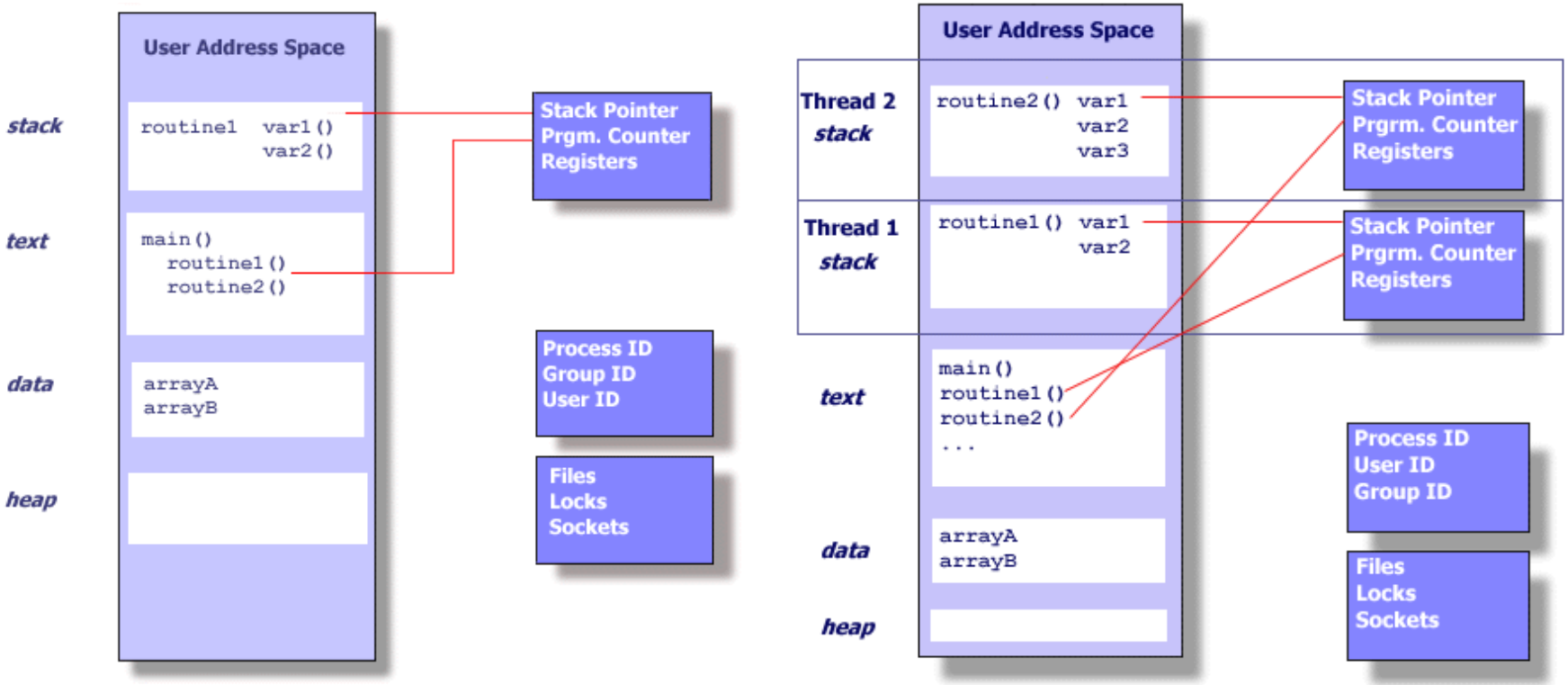
```
for (row = 0; row < n; row++)
  for (column = 0; column < n; column++)
        c[row][column] =
        create_thread( dot_product(get_row(a, row),
                       get_col(b, col)));
```

Note that there're no any dependences among dot_product() thus we can create multiple threads to solve them concurrently.

# Thread Basics

- All address space in the logical machine model of a thread is globally accessible to every thread.
- The stack corresponding to the function call is generally treated as being local to the thread.
- This implies a logical machine model with both global memory (default) and local memory (stacks).
- For NUMA machine, it is important to note that such a flat model may result in poor performance since memory is physically distributed in typical machines
  - Locality matters

# Thread Basics




Unix process

Threads within a Unix process

Source: https://computing.llnl.gov/tutorials/pthreads/

# The POSIX Thread API

- Commonly referred to as Pthreads, POSIX has emerged as the standard threads API, supported by most vendors
  - IEEE standard 1003.1c-1995 POSIX API
  - Vendors may have vendor-specific APIs
- The concepts discussed here are largely independent of the API
  - Can be used for programming with other thread APIs
  - NT threads, Solaris threads, Java threads, etc.

# Pthreads Creation and Termination

- Pthreads creation:

```
#include <pthread.h>
int pthread_create (
   pthread_t *thread_handle, const pthread_attr_t
     *attribute,
   void * (*thread_function)(void *),
   void *arg);
```

- The function pthread_create invokes function thread_function as a thread

# Pthreads Creation and Termination

- Pthreads termination

```
int pthread_join (
   pthread_t thread,
   void **ptr);
```

Waits for the termination of thread with id given by the thread variable (often called by "master thread")

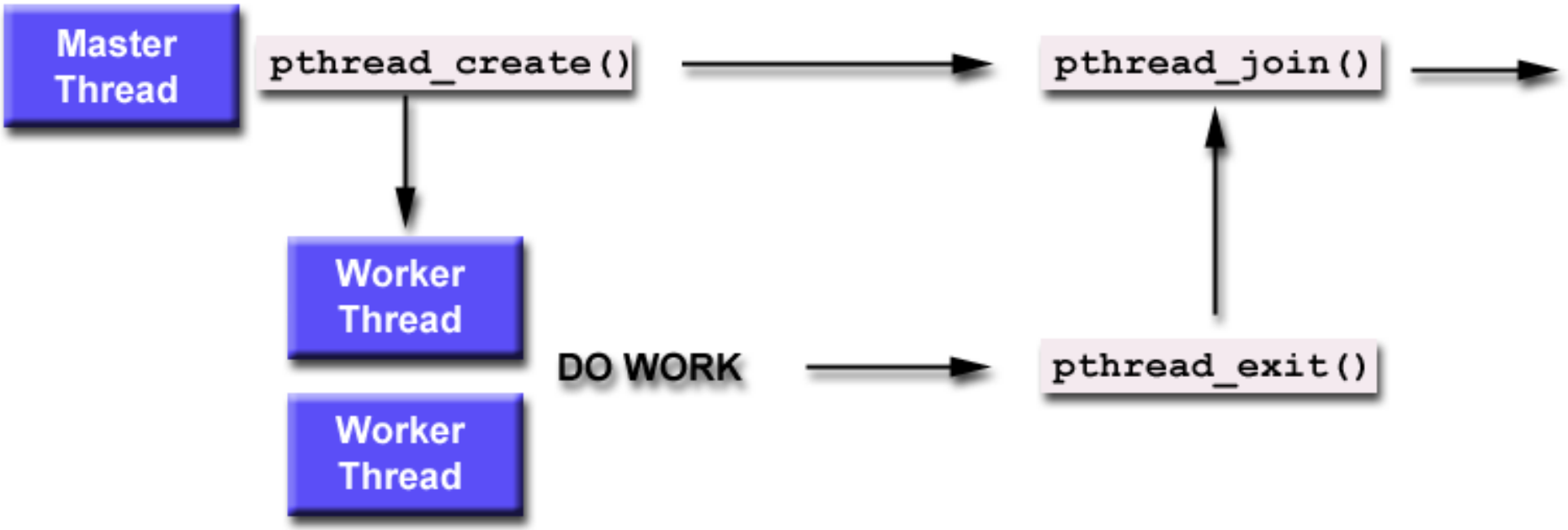
```
void pthread_exit(void *value_ptr);
```

Terminates the calling thread and makes the value value_ptr available to any successful join (often called by "worker thread")

```
int pthread_cancel(pthread_t thread);
```

Can be cancelled by another thread

# Pthreads Creation and Termination



# Pthreads Example Code: Hello World

Need to include the header file

Define a function to be run by threads

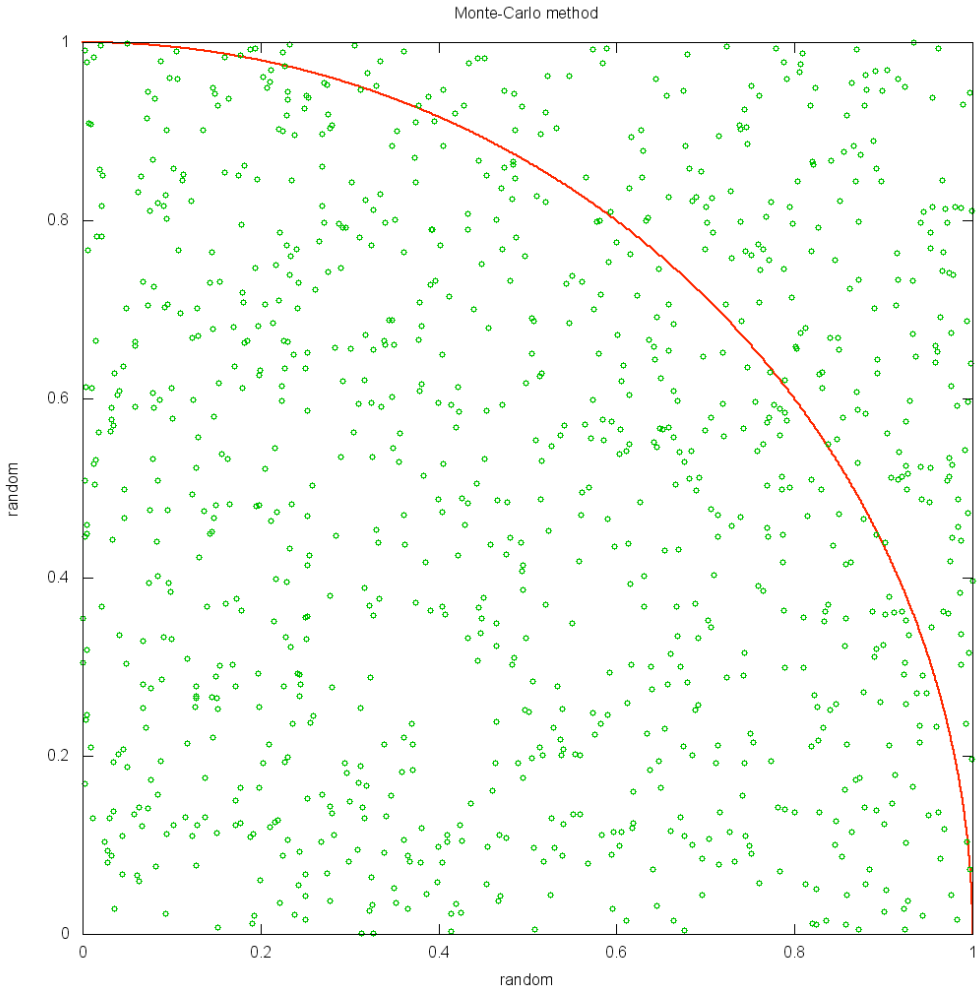
Define threads

Create threads and run with the specified function

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define NUMBER_OF_THREADS     10

void *print_hello_world(void *tid)
{
        /* This function prints the thread's identifier and then exits. */
        printf("Hello World. Greetings from thread %d0, tid);
        pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
        /* The main program creates 10 threads and then exits. */
        pthread_t threads[NUMBER_OF_THREADS];
        int status, i;

        for(i=0; i < NUMBER_OF_THREADS; i++) {
                printf("Main here. Creating thread %d0, i);
                status = pthread_create(&threads[i], NULL, print_hello_world, (void *)i);

                if (status != 0) {
                        printf("Oops. pthread_create returned error code %d0, status);
                        exit(-1);
                }
        }
        exit(NULL);
}
```

# Pthreads Example Code: Compute pi

- Generating random points in a unit length square
- Counting the number of points fall within circle
- The faction of random points: pi/4



# Questions?

## Questions/Suggestions/Comments are always welcome!

Write me: yong.chen@ttu.edu
Call me: 806-834-0284
See me: ENGCTR 315

*If you write me an email for this class, please start the email subject with [CS4379] or [CS5379].*