

# Parallel/GPU Computing and HPC Research: Past, Present and Future



Research Conclave 2025 Talk @ TCE

\*\*\*

Rajesh Pandian M

[www.mrprajesh.co.in](http://www.mrprajesh.co.in)

# Outline

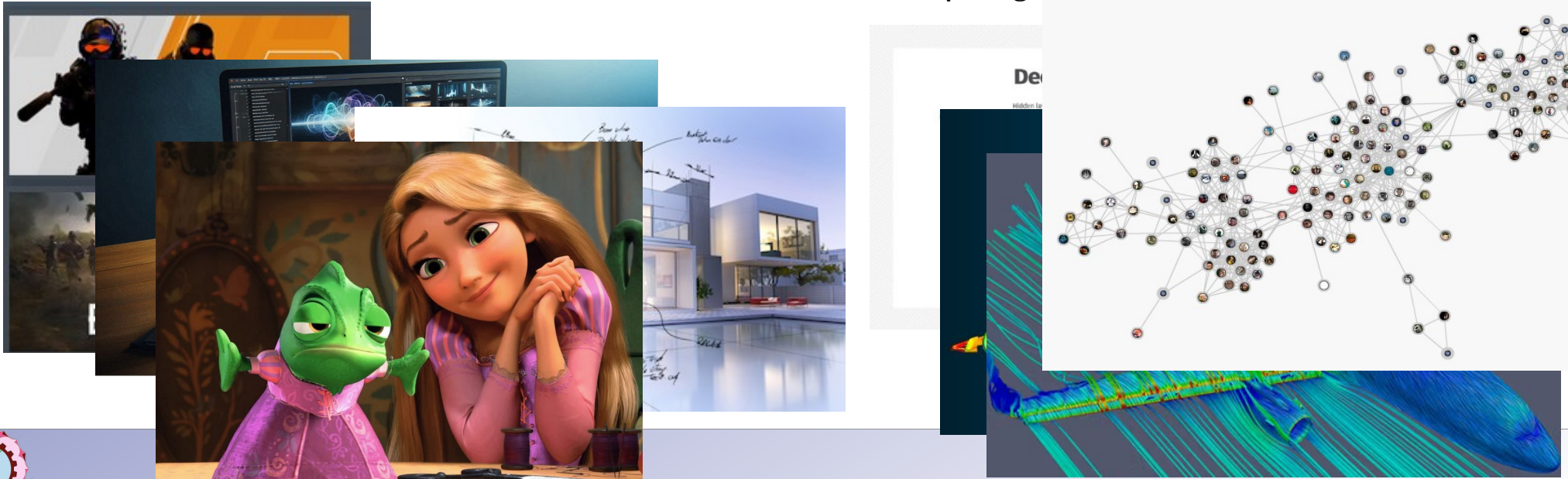
---

- Motivation
- Beginning of GPGPU era
  - It all started here in India/IIT
- Present
  - Graph algorithms
  - AL/ML Training
- Future
- Summary



# Motivation

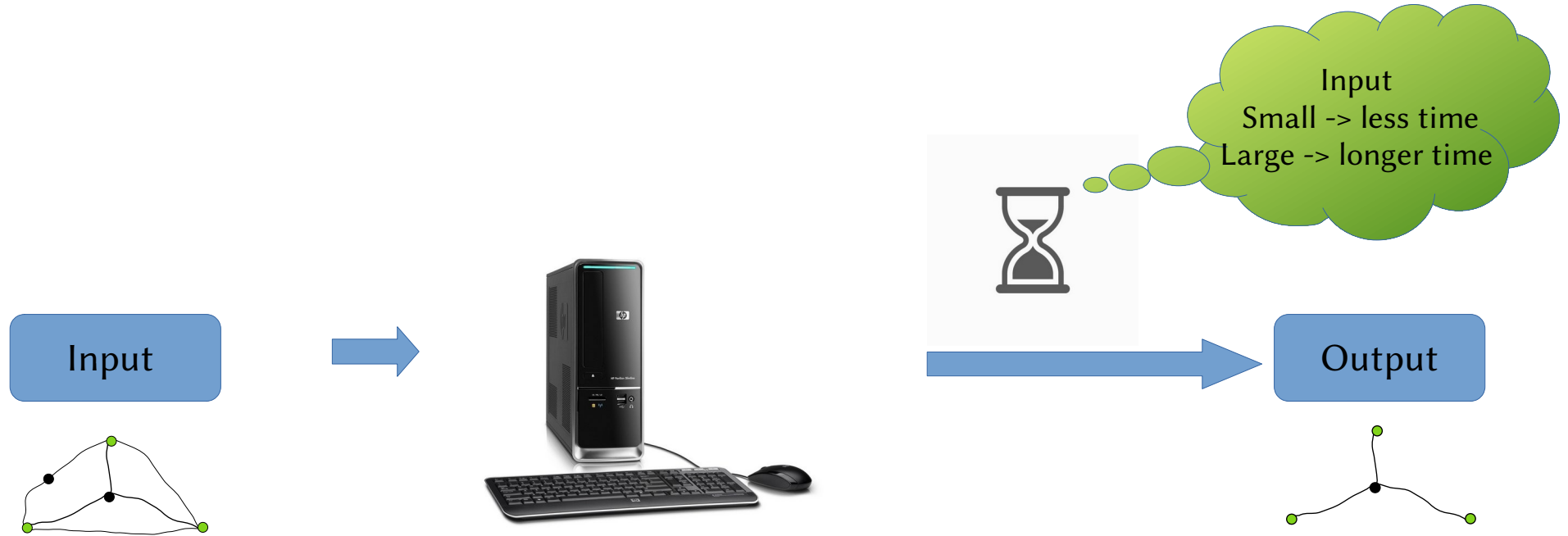
- Graphics
  - 3D Games
  - Video rendering
  - 3D Graphics
  - Animations
- Non-Graphics
  - Machine learning
  - Scientific simulations
  - HPC
  - Graph algorithms



2016. Joel Hruska 10 years ago, Geforce 8800 graphics card changed PC gaming, computing forever



# Sequential Computing



Graph

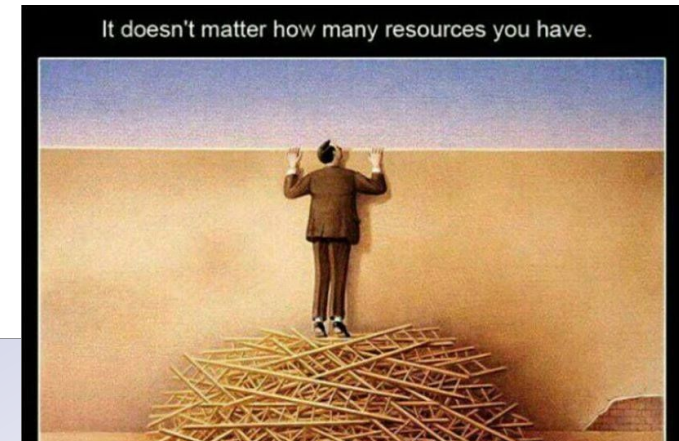
- Minimum Spanning Tree
- Single source shortest path



# Parallel Computation



Top 10 super computers. 7/10 GPUs



# Parallel Platforms

Software/Frameworks

OpenMP

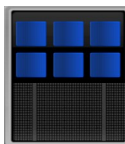
ROCm

MPI

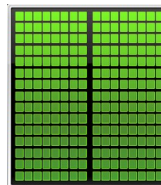
OpenACC

OpenCL

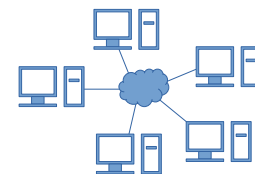
Underlying hardware for parallelization



CPU



GPU



Distributed



# The beginning of GPGPU

---

## Accelerating large graph algorithms on the GPU using CUDA

Pawan Harish and P. J. Narayanan

Center for Visual Information Technology  
International Institute of Information Technology Hyderabad, INDIA  
{harishpk@research., pjn@}iiit.ac.in

HiPC'2007

**Abstract.** Large graphs involving millions of vertices are common in many practical applications and are challenging to process. Practical-time implementations using high-end computers are reported but are accessible only to a few. Graphics Processing Units (GPUs) of today have high computation power and low price. They have a restrictive programming model and are tricky to use. The G80 line of Nvidia GPUs can be treated as a SIMD processor array using the CUDA programming model. We present a few fundamental algorithms – including breadth first search, single source shortest path, and all-pairs shortest path – using CUDA on large graphs. We can compute the single source shortest path on a 10 million vertex graph in 1.5 seconds using the Nvidia 8800GTX GPU costing \$600. In some cases optimal sequential algorithm is not the fastest on the GPU architecture. GPUs have great potential as high-performance co-processors.





# Why GPU for graph processing?

- GPU for General Purpose = GPGPU

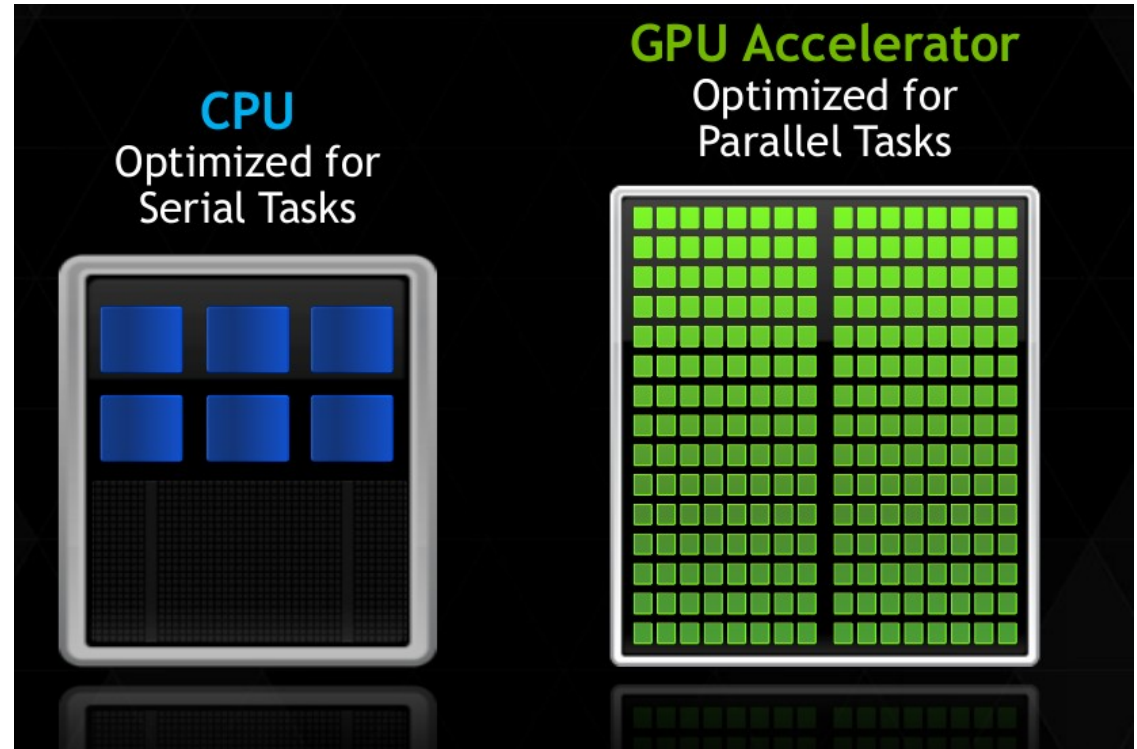


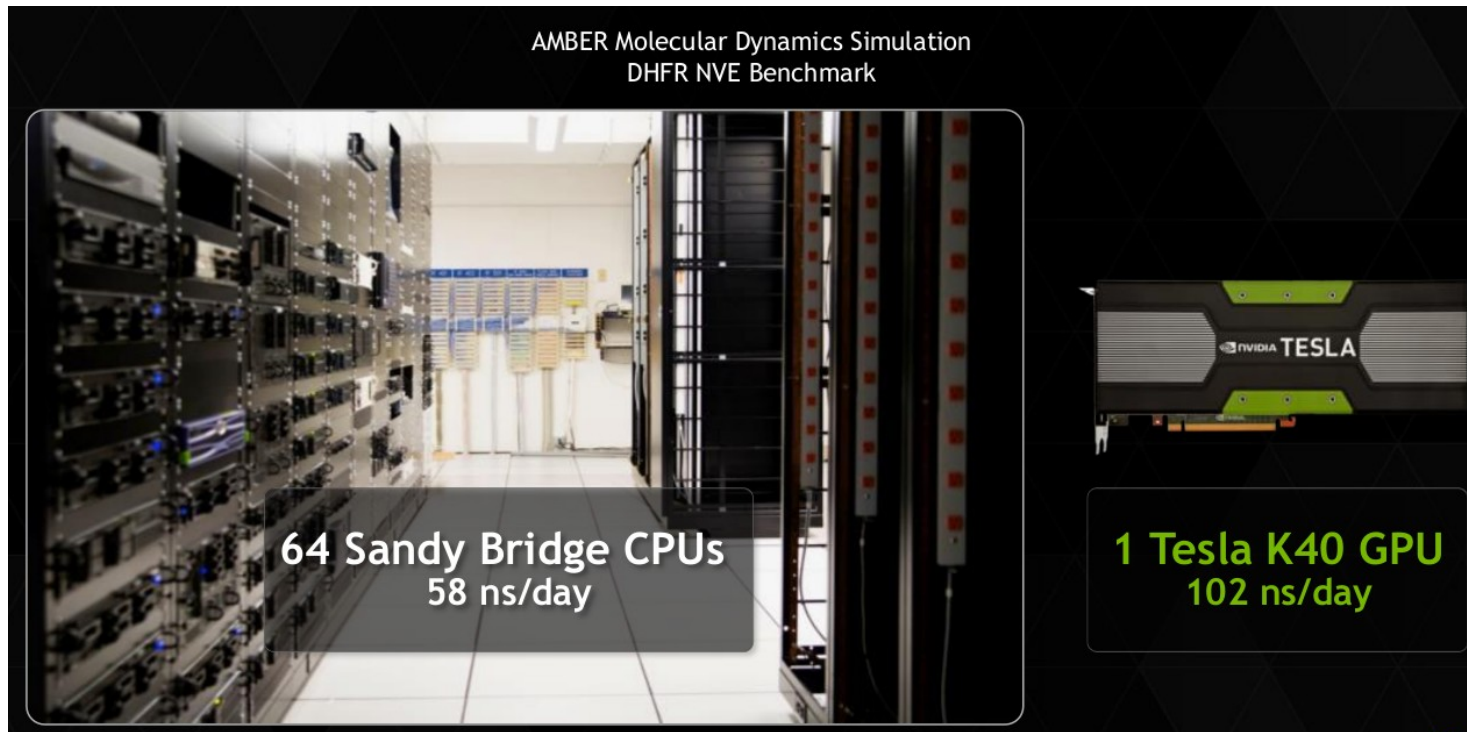
Image Source: GPU Technology Conference (GTC) 2014





# Why GPU for graph processing?

- GPU for General Purpose = GPGPU
- **Computation power**



Source: GPU Technology Conference (GTC) 2014  
Harvey et al. *Journal of Chemical Theory and Computation*, 2009

# Why GPU for graph processing?

- GPU for General Purpose = GPGPU
- Computation power
- Lesser cost
- Energy efficient compared with CPUs

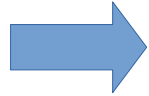


Image Source: GPU Technology Conference (GTC) 2014

# Graphs are everywhere

- Networks: Road, Social, Biological, Wireless, and more.
- Such Graphs are extremely large.
- Faster computation is expected.

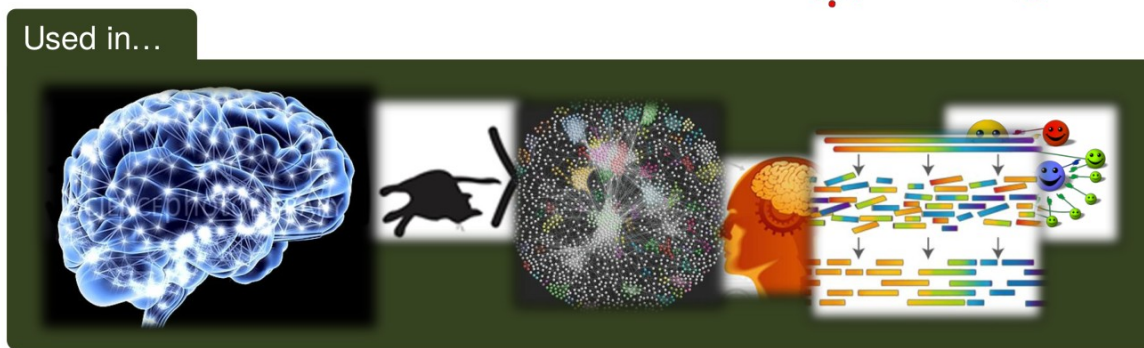
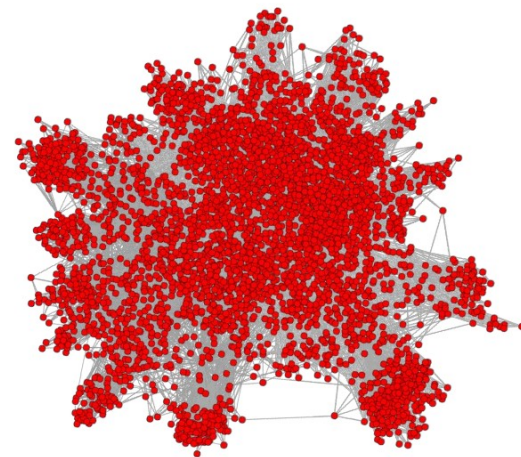
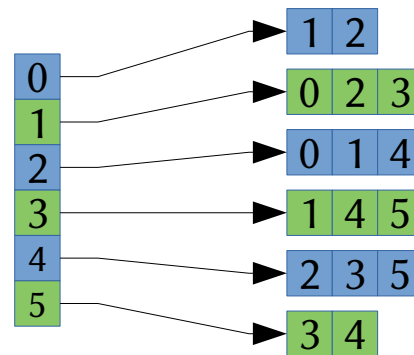
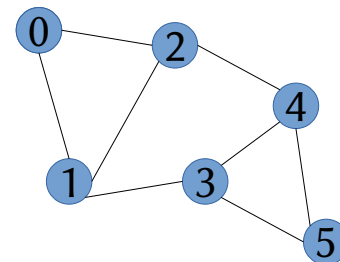


Image Source: <https://spcl.inf.ethz.ch/Publications/.pdf/pushpull-slides.pdf>

# Graph Representation

- Adjacency Matrix
- Adjacency list

0	0	1	1	0	0	0
1	1	0	1	1	0	0
2	1	1	0	0	1	0
3	0	1	0	0	1	1
4	0	0	1	1	0	1
5	0	0	0	1	1	0
	0	1	2	3	4	5



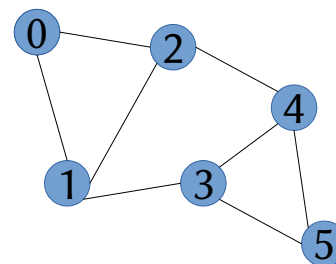
# Graph Representation – EdgeList

- **EdgeList / Coordinate Format (COO)**
- Compressed Spare Row (CSR)

Reduce repetitions?

0 0 1 1 1 2 2 2 3 3 3 4 4 4 5 5

1 2 0 2 3 0 1 4 1 4 5 2 3 5 3 4

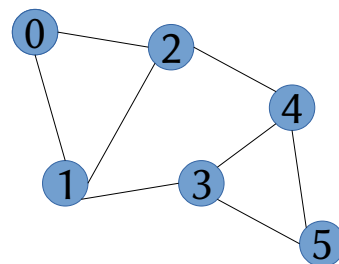
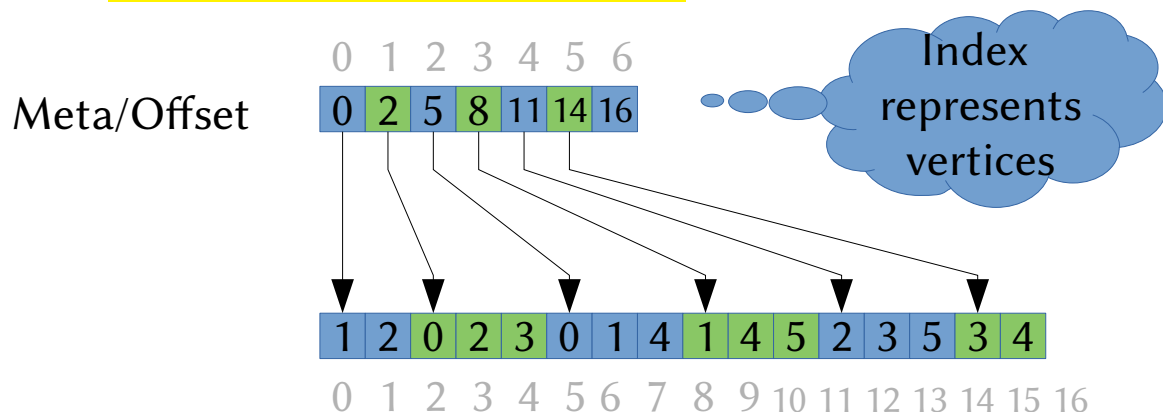


Two 2m-sized array



# Graph Representation - CSR

- EdgeList / Coordinate Format (COO)
- **Compressed Spare Row (CSR)**



**Two arrays:**

Offset array :  $n+1$

CSRList :  $2m$

<https://networkrepository.com/mtx-matrix-market-format.html>





# Operator-formulation approach

```
Initialize(G,...);  
do {  
    Operator-Apply(G,...);  
    // on set of nodes, edges  
    // or property of G  
} while (condition);
```

Parallel

Operator = GPU kernel

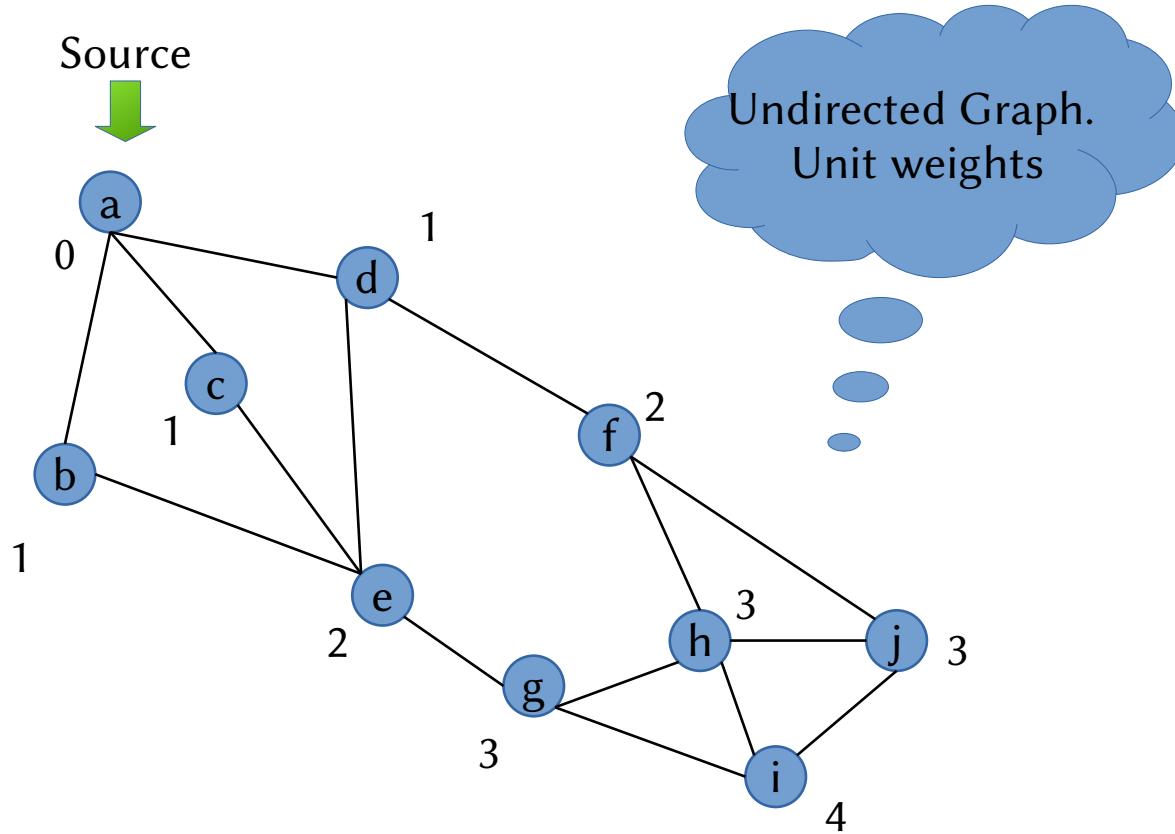
- BFS // Computing level information from source
- SSSP // Propagating distance from source



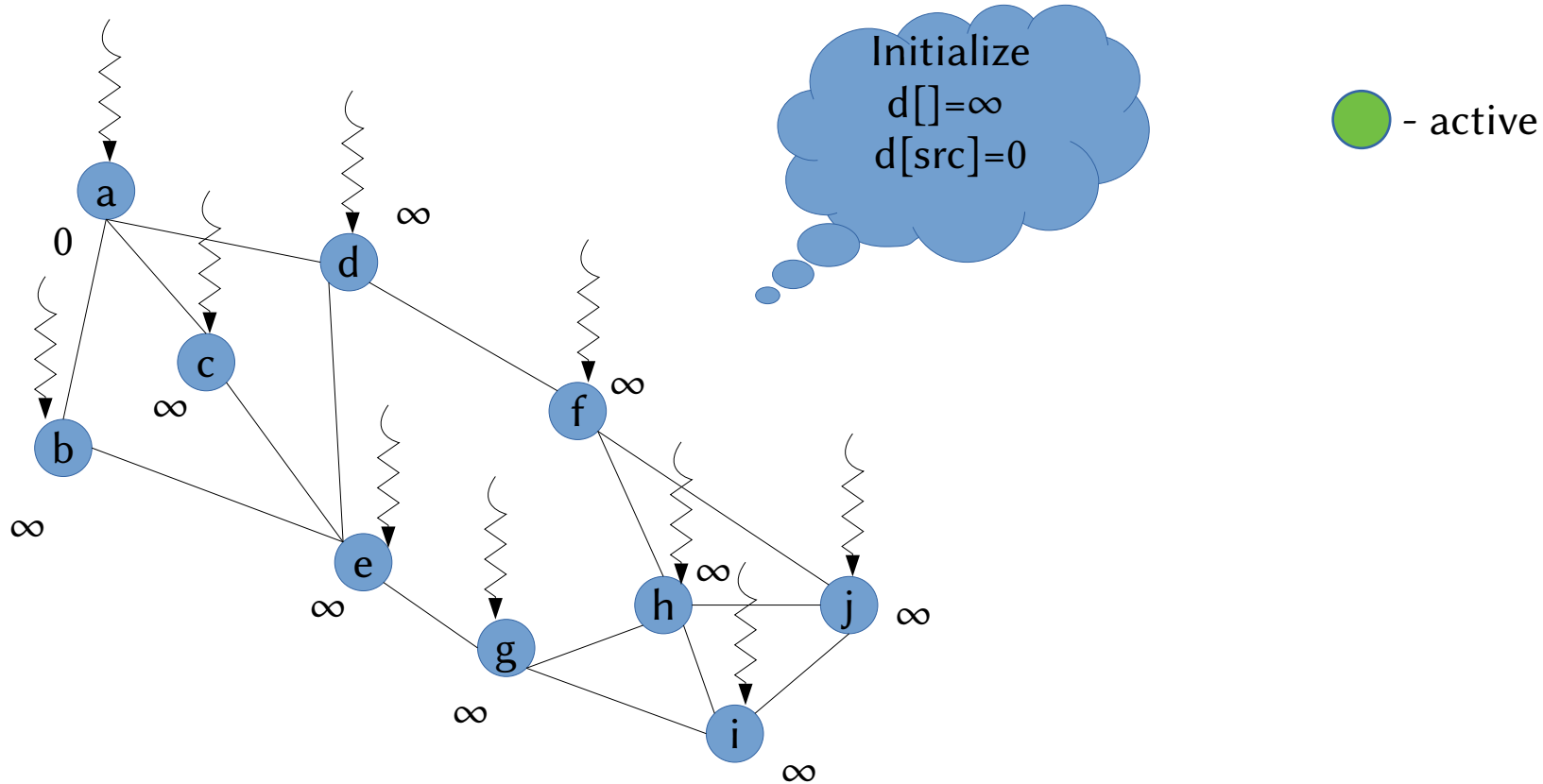
Pingali et. al. *The Tao of parallelism in algorithms*. PLDI, 2011



# BFS/SSSP



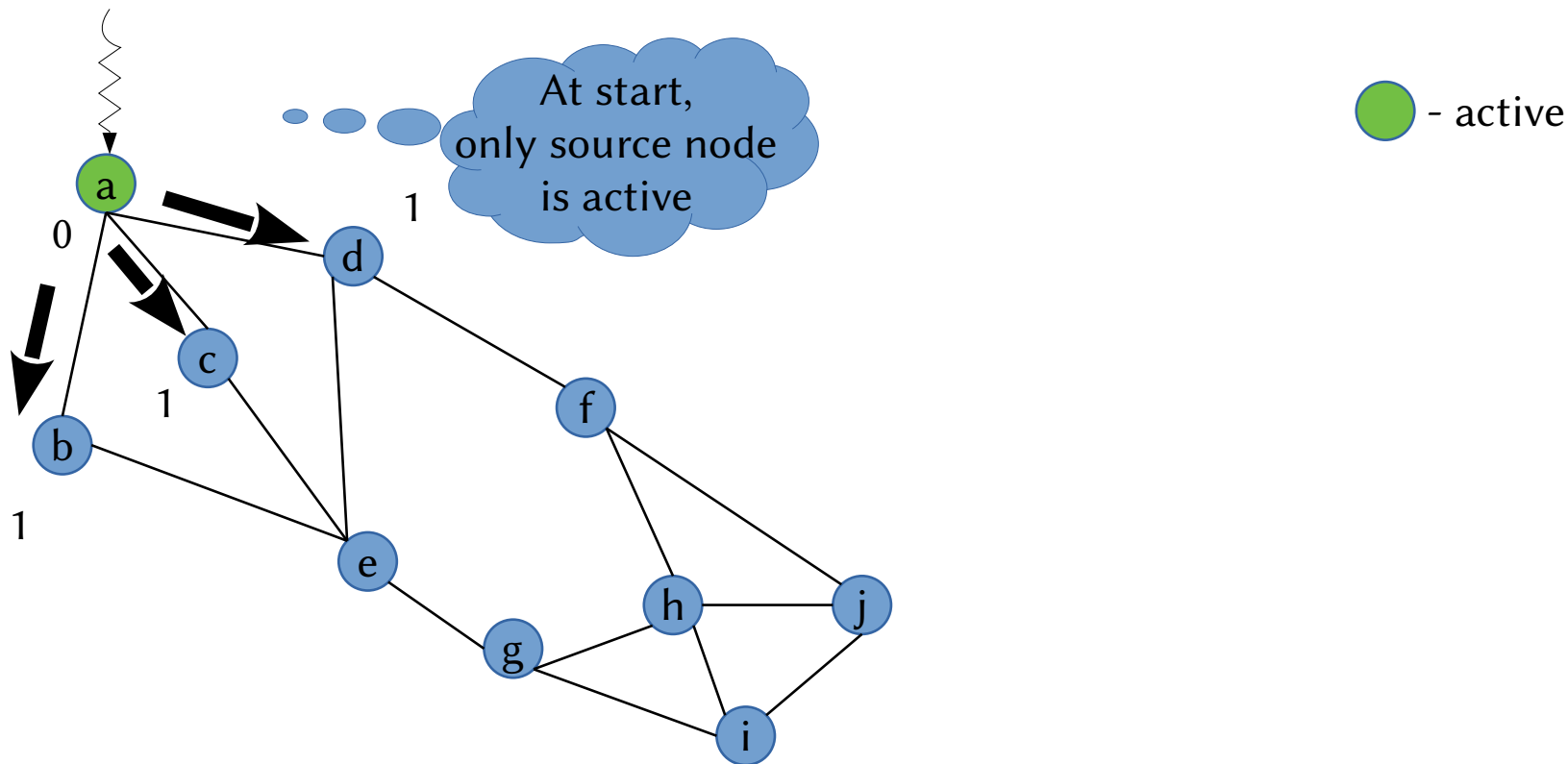
# SSSP: Single Source Shortest Path



Rupesh Nasre; Martin Burtscher; Keshav Pingali.

*Data-Driven Versus Topology-driven Irregular Computations on GPUs*, IPDPS 2013.

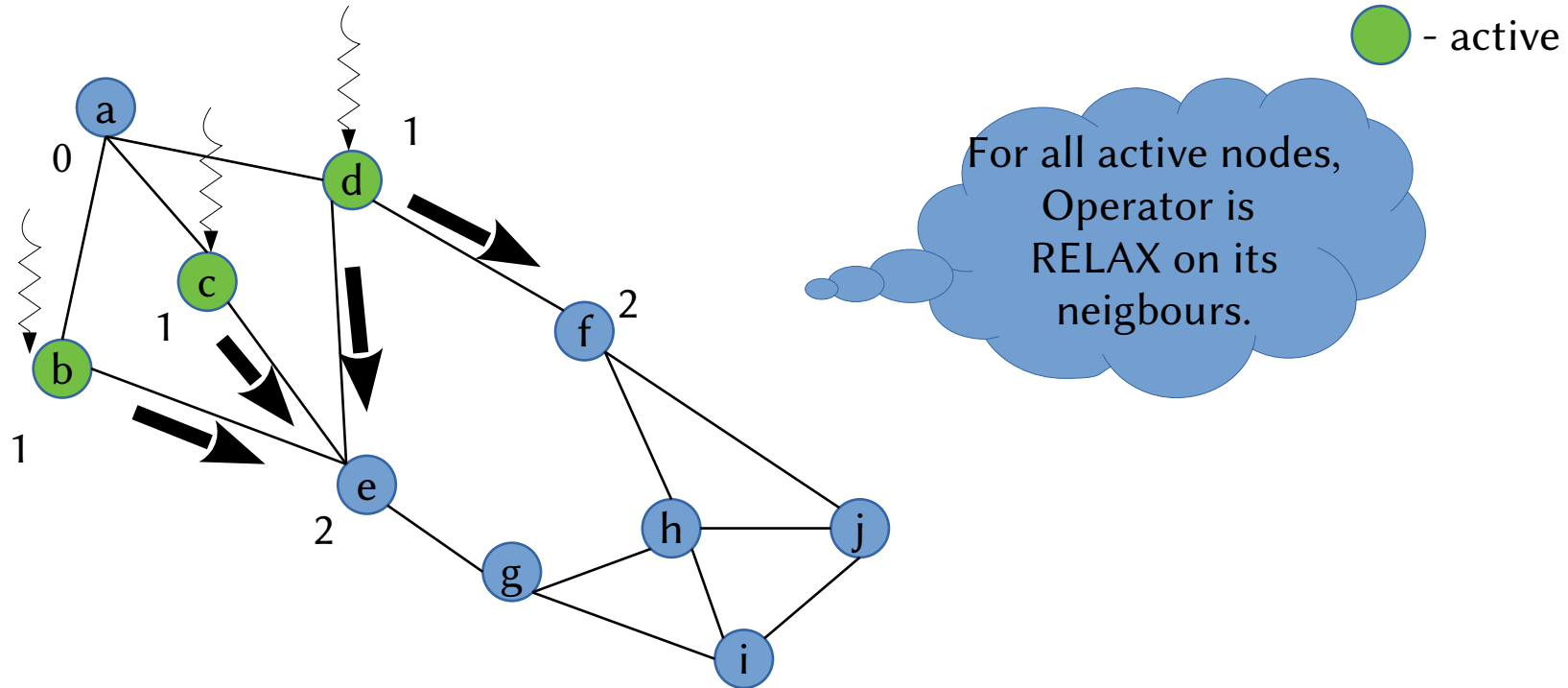
# SSSP: Data-driven



Rupesh Nasre; Martin Burtscher; Keshav Pingali.

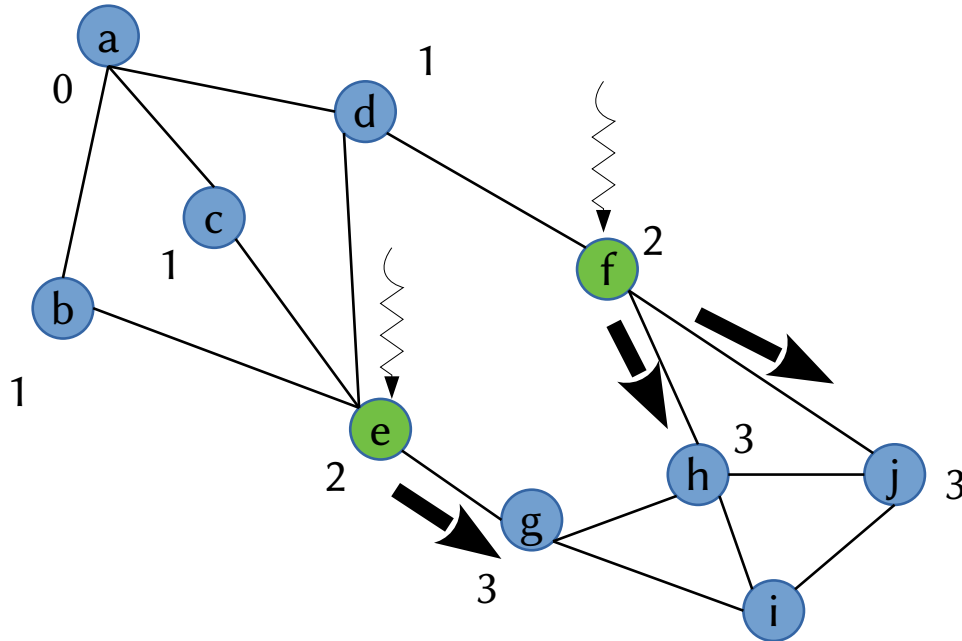
*Data-Driven Versus Topology-driven Irregular Computations on GPUs*, IPDPS 2013.

# SSSP: Data-driven



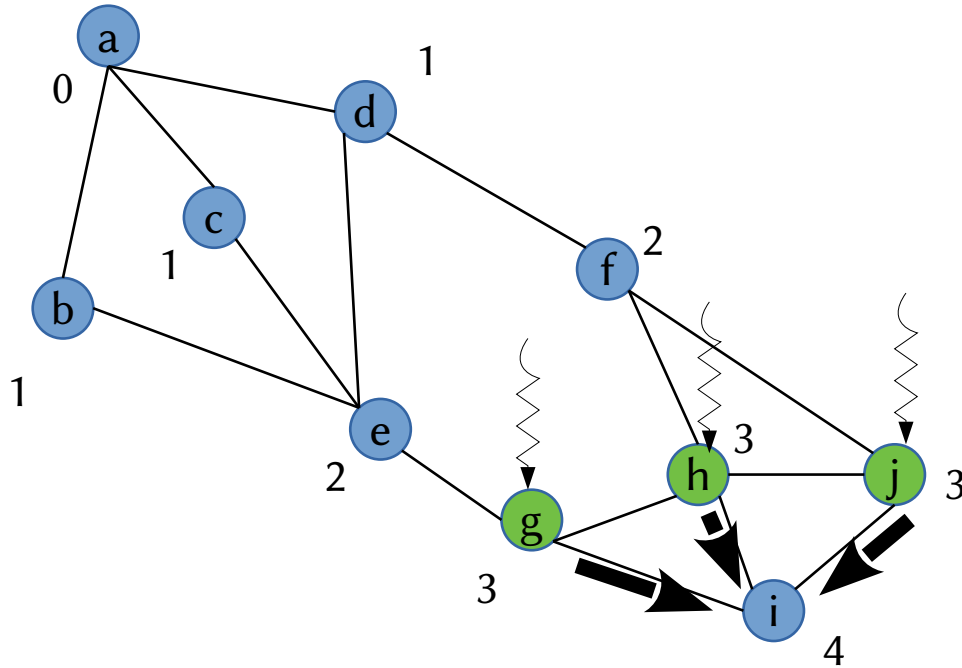
# SSSP: Data-driven

● - active



# SSSP: Data-driven

● - active

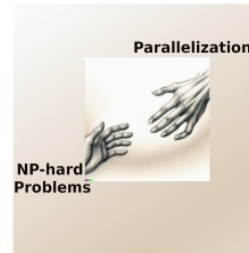


# PhD Thesis



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY MADRAS  
CHENNAI – 600036

## NP-hard Problems meet Parallelization



*A Thesis*

*Submitted by*

**RAJESH PANDIAN M**





# Our Philosophy

... take a **fresh look** at some of the classic graph algorithms and devise **faster** and more parallel GPU and CPU implementations.

+

NP-hard

=

Our Philosophy

- Fallin et al.

## A High-Performance MST Implementation for GPUs

Alex Fallin  
Dept. of Computer Science  
Texas State University  
San Marcos, Texas, USA  
waf13@txstate.edu

Andres Gonzalez  
Dept. of Computer Science  
Texas State University  
San Marcos, Texas, USA  
ag1548@txstate.edu

Jarim Seo  
Dept. of Computer Science  
Texas State University  
San Marcos, Texas, USA  
j\_s1195@txstate.edu

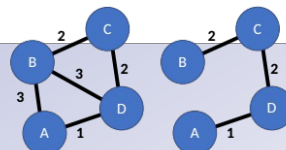
Martin Burtscher  
Dept. of Computer Science  
Texas State University  
San Marcos, Texas, USA  
burtscher@txstate.edu

SC'23

### ABSTRACT

Finding a minimum spanning tree (MST) is a fundamental graph algorithm with applications in many fields. This paper presents ECL-MST, a fast MST implementation designed specifically for GPUs. ECL-MST is based on a parallelization approach that unifies Kruskal's and Borůvka's algorithm and incorporates new and existing optimizations from the literature, including implicit path compression and edge-centric operation. On two test systems, it outperforms leading GPU and CPU codes from the literature on all of our 17 input graphs from various domains. On a Titan V GPU,

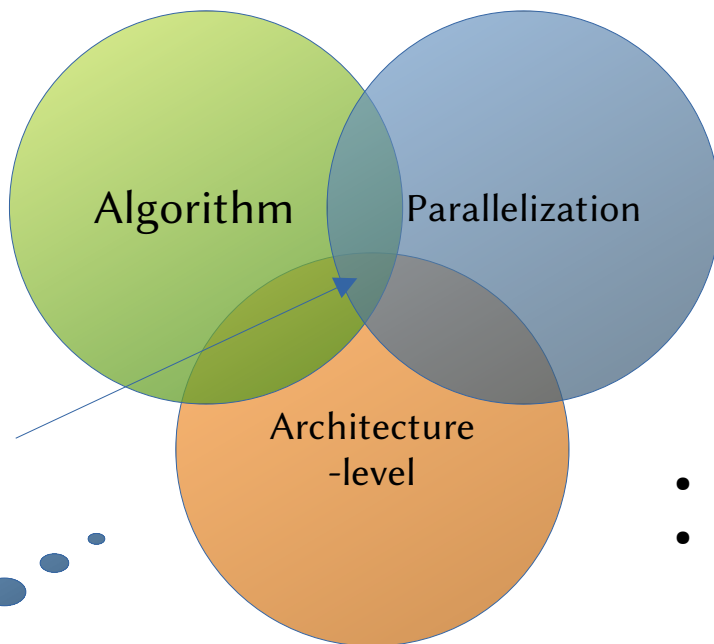
lines. In this example, the cheapest distribution grid that allows everyone to deliver or receive electricity is the MST shown.



# Optimizing for peak performance

## OPTIMIZATIONS

- Input Characteristics
- Properties of substeps



- Vertex-/Edge- centric
- Data-/Topology-driven
- Push-/Pull-based

...

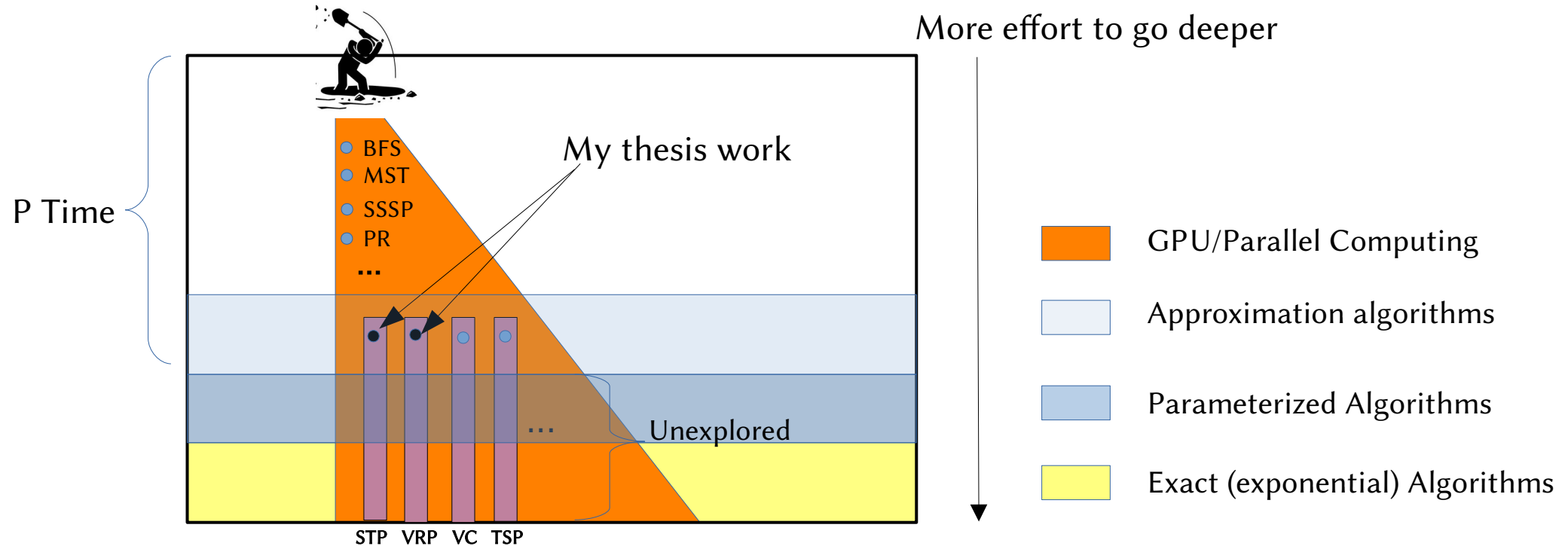
Even more  
Peak performance

- Shared memory
- Warp-Intrinsics

Is that all?

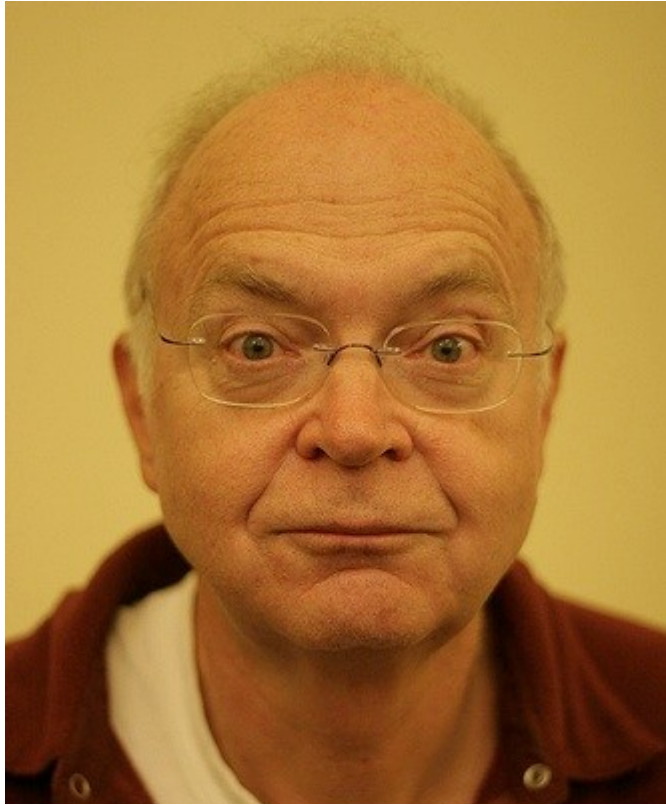


# Landscape of Parallelization



# Communicating Research

---



“Of equal importance to  
**solving a problem** is the **communication**  
of that **solution to others.**”  
- Donald Knuth



# Summary

---

- Parallel Computing is here to stay forever
- Make your program to run in parallel
- Innovative ideas are more valued than tiny delta ideas
- Efficient algorithms and optimizations make faster parallel programs
- GPUs are everywhere, use it wisely.
- Efficiency ----> Sustainability.

<https://mrprajesh.github.io/pages/research.html>

**Thank you.**

