CAPC

CDAC's Automatic Parallelizing Compiler

Prachi Pandey
System Software Development Group
C-DAC Bangalore

World has moved to Multicores!

- All latest systems - servers, desktops, laptops, mobiles have **multi/many cores**
- A program should run on **all the cores simultaneously to speed up** the execution time.
- ***Parallelism is the need of the hour!***
- **But** most of the programs are still serial and run on single core

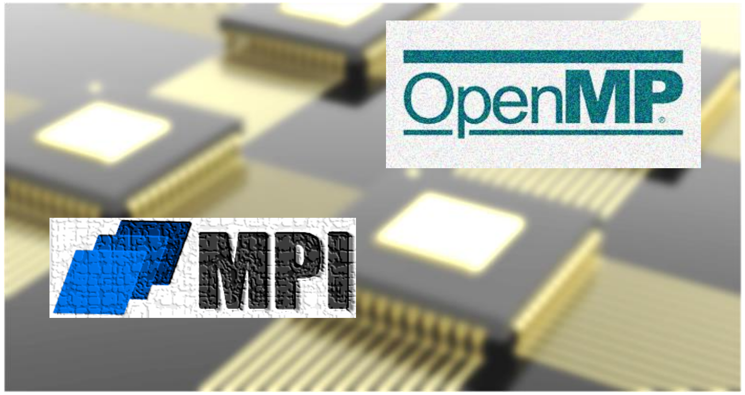
VS.
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] \ELI\CPP_ST~1\OLD_C_~1\BORLAND\MYPROGS\FIBONACI.C 3=[↕]
1:1
Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

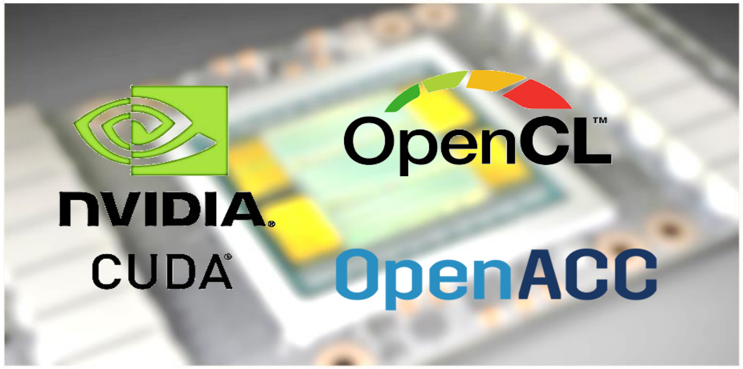
```
#include <stdio.h>
#include <conio.h>

int i, j, inpt;
ar[20];

main()
{
  clrscr();

  printf("Enter [illegible]mber (1 to 20) ? ");
  scanf("%d",&[illegible]pt);

  ar[0] = a[illegible]] = 1;
  printf([illegible]1 1");

  for[illegible] 2; i <= inpt; i++)
  {
  [illegible]i] = ar[i-1] + ar[i-2];
  [illegible]intf(" %d",ar[i]);
```

Motivation

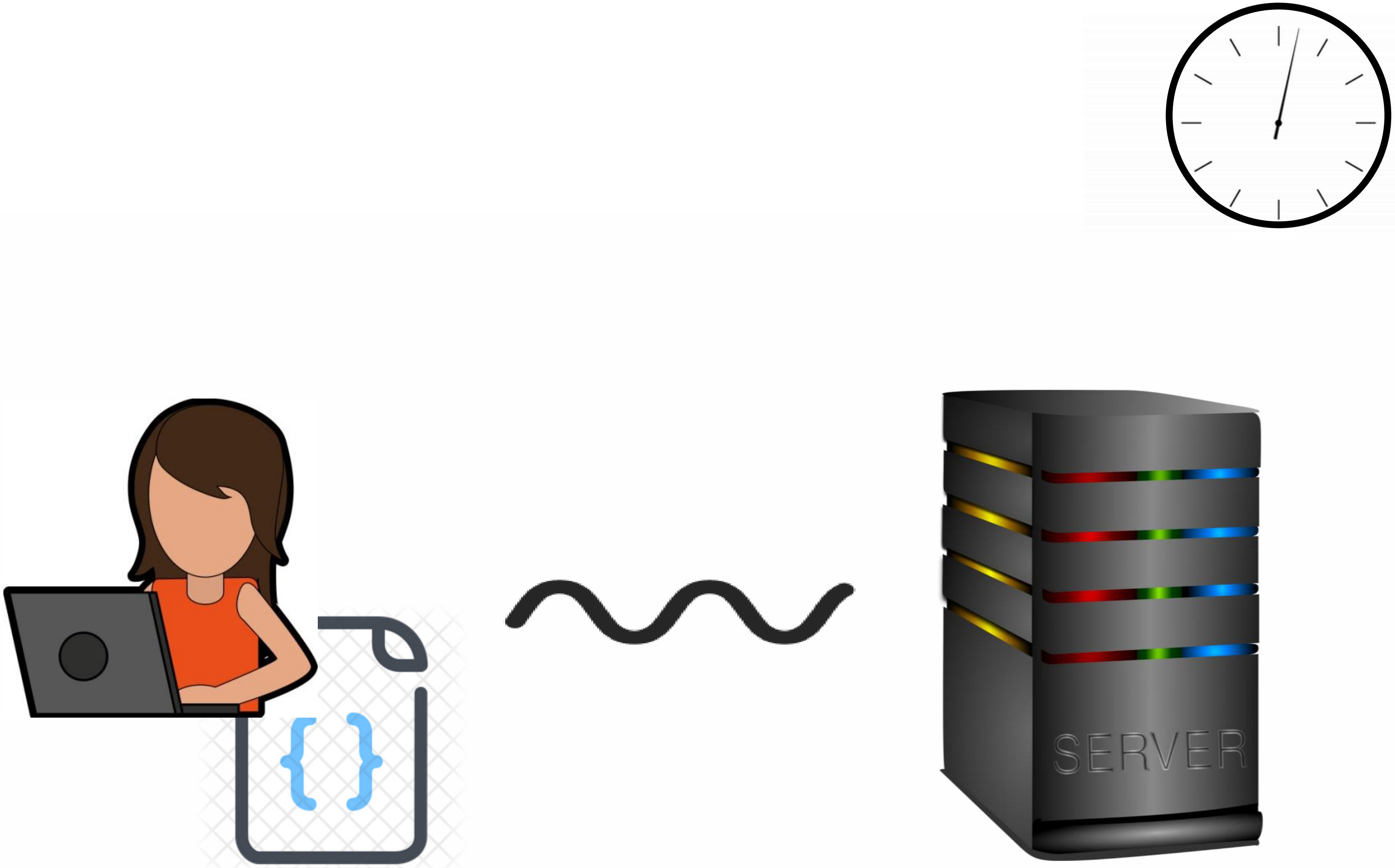
Parallel Programming is still considered complex and high end

No single language for different parallel hardware

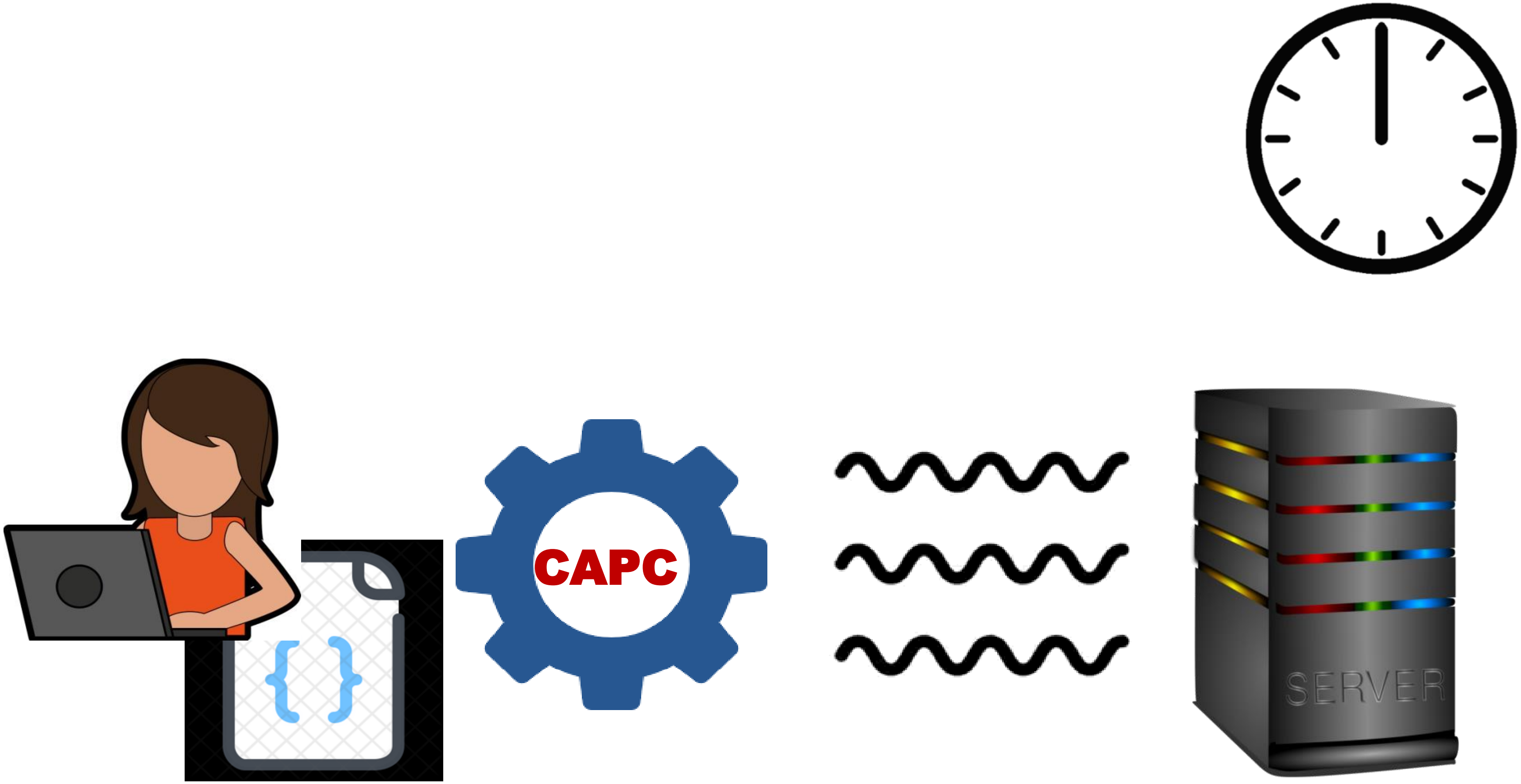




"I wish my sequential Program could run as it is on the latest Parallel Hardware!"



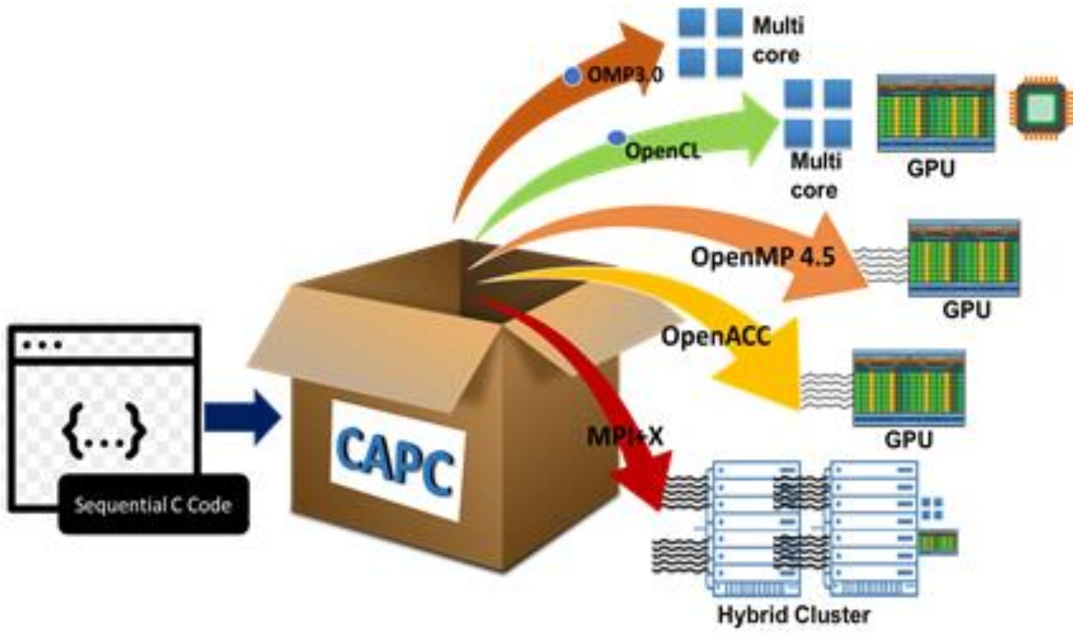
Sequential Program Execution



Parallel Program Execution

CAPC

C-DAC's **A**utomatic **P**arallelizing **C**ompiler, CAPC, automatically converts sequential programs to the equivalent parallel programs for target parallel architectures



How to use CAPC

- Copy the folder CAPC2.0 to your home directory
- Open the file “env.sh”, modify “CAPC_HOME” environment variable to point to your home directory ($PWD)
- Execute the command

  ```
  source env.sh
  ```

- To parallelize your C code for multicores, execute the following command

  ```
  capc –c2omp <inputcode.c>
  ```

Features

- Ease of use
- Automatic parallelization
- Human readable output
- Support for multiple Parallel paradigms
- Profitability estimate
- Vendor agnostic

Benefits

Automatic Parallelization of legacy codes

Improves programmer productivity

Speedup for large applications

Jumpstart Parallel programming

Performance Evaluation

Below table represents the speedup obtained after parallelization through CAPC. The experiments have been performed on our test machine which has similar architecture and configuration as PARAM Shakti.

Sl. No	Application Name	Input data size	Serial execution time (in secs)	Execution time after parallelization (24 cores)	Speedup obtained
1	Matrix multiplication	1000	11.226	1.064	10.55X
2	Monte Carlo PI calculation	500 million	5.765	0.778	7.4X
3	Jacobi-2D equation	1000	0.024	0.006	4X
4	Heat-2D equation	500	5.071	0.553	9.16X
5	Hill cipher	13000	1.349	0.289	4.67X
6	Symmetric	30000	24.828	3.410	7.28X
7	Compression	30000	11.810	1.522	7.76X

Performance Evaluation

Sl. No	Application Name	Input data size	Serial execution time (in secs)	GPU Execution time after parallelizat-ion (NVIDIA V100)	Speedup obtained
1	Matrix multiplication	2K x 2K	1.38	0.0144	95X
2	Heat-3D equation	500 x 500	4.95	1.655	3X
3	Matrix Transpose	8K x 8K	2200	300	7.33X
4	3D-Matrix arithmetic	150 x 150	0.050	0.0483	1.15X
5	Jacobi-2D equation	18K x 18K	19	11.5	1.7X

Questions ??

THANK YOU!