University of Zurich[UZH]

High Performance Computing Lecture 3

Douglas Potter
Jozef Bucko
Noah Kubli

Where are we?

We can run codes

- Submit on a SLURM queue

We can compile codes

- More on this today

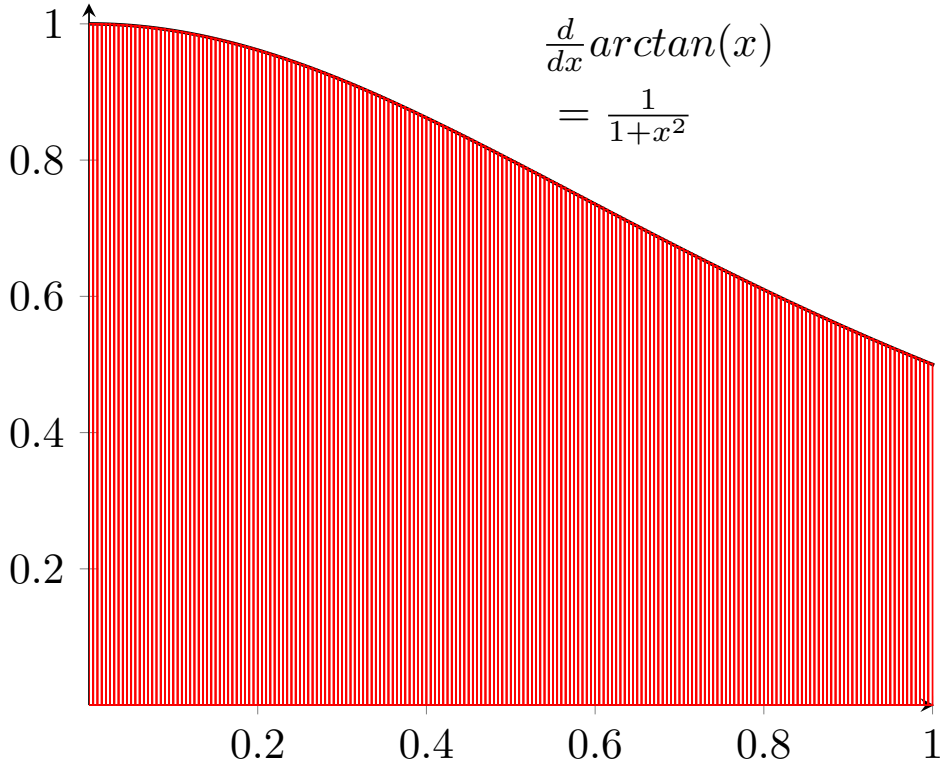
Understanding Parallel

- Main topic today

GOAL: change a code

- Maybe write your own

Exercise: Compute PI in Parallel



Integration in Python

```
N=5
dx=1.0 / N
s = 0
for x in range(0, N):
  s += dx / ( 1 + ((x+0.5)*dx)**2 )
print(s*4)

dhcp-94-191:cpi$ python integrate.py
3.1449258640003328
```

University of Zurich[UZH]

OpenMP Results (multi-threaded) N=1,000,000,000

```
running on 1 threads: PI = 3.141592653589971 computed in 4.55 seconds
running on 2 threads: PI = 3.141592653589901 computed in 2.291 seconds
running on 3 threads: PI = 3.141592653589962 computed in 1.624 seconds
running on 4 threads: PI = 3.141592653589821 computed in 1.258 seconds
running on 5 threads: PI = 3.141592653589596 computed in 1.041 seconds
running on 6 threads: PI = 3.141592653589682 computed in 0.8988 seconds
running on 7 threads: PI = 3.14159265358963 computed in 0.7989 seconds
running on 8 threads: PI = 3.141592653589769 computed in 0.7217 seconds
running on 9 threads: PI = 3.141592653589656 computed in 0.6415 seconds
running on 10 threads: PI = 3.141592653589794 computed in 0.5774 seconds
running on 11 threads: PI = 3.14159265358966 computed in 0.5249 seconds
running on 12 threads: PI = 3.14159265358986 computed in 0.4811 seconds
running on 13 threads: PI = 3.141592653589865 computed in 0.4441 seconds
running on 14 threads: PI = 3.141592653589788 computed in 0.4124 seconds
running on 15 threads: PI = 3.141592653589805 computed in 0.3849 seconds
running on 16 threads: PI = 3.141592653589832 computed in 0.3609 seconds
running on 17 threads: PI = 3.141592653589839 computed in 0.3397 seconds
running on 18 threads: PI = 3.141592653589814 computed in 0.3208 seconds
running on 19 threads: PI = 3.141592653589826 computed in 0.3053 seconds
running on 20 threads: PI = 3.141592653589855 computed in 0.2897 seconds
running on 21 threads: PI = 3.141592653589775 computed in 0.2768 seconds
running on 22 threads: PI = 3.141592653589823 computed in 0.2644 seconds
running on 23 threads: PI = 3.141592653589866 computed in 0.2528 seconds
running on 24 threads: PI = 3.141592653589792 computed in 0.2423 seconds
running on 25 threads: PI = 3.14159265358978 computed in 0.2326 seconds
running on 26 threads: PI = 3.141592653589832 computed in 0.2237 seconds
running on 27 threads: PI = 3.141592653589835 computed in 0.2154 seconds
running on 28 threads: PI = 3.141592653589816 computed in 0.2077 seconds
running on 29 threads: PI = 3.141592653589819 computed in 0.2006 seconds
running on 30 threads: PI = 3.141592653589893 computed in 0.1939 seconds
running on 31 threads: PI = 3.141592653589744 computed in 0.1876 seconds
running on 32 threads: PI = 3.141592653589758 computed in 0.1818 seconds
running on 33 threads: PI = 3.141592653589806 computed in 0.1763 seconds
running on 34 threads: PI = 3.141592653589926 computed in 0.1711 seconds
running on 35 threads: PI = 3.14159265358979 computed in 0.1662 seconds
running on 36 threads: PI = 3.141592653589822 computed in 0.1616 seconds
```

```
#!/bin/bash -l
#SBATCH --account=uzh8
#SBATCH --job-name=openmp_test
#SBATCH --time=01:00:00
#SBATCH --nodes=(nodes)
#SBATCH --ntasks-per-node=(processes)
#SBATCH --cpus-per-task=(threads)
#SBATCH --partition=normal
#SBATCH --constraint=mc

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

srun cpi_openmp
```

University of Zurich[UZH]

OpenMP Results (multi-threaded) N=1,000,000,000

```
running on 1 threads: PI = 3.141592653589971 computed in 4.55 seconds
running on 2 threads: PI = 3.141592653589901 computed in 2.291 seconds
running on 3 threads: PI = 3.141592653589962 computed in 1.624 seconds
running on 4 threads: PI = 3.141592653589821 computed in 1.258 seconds
running on 5 threads: PI = 3.141592653589596 computed in 1.041 seconds
running on 6 threads: PI = 3.141592653589682 computed in 0.8988 seconds
running on 7 threads: PI = 3.14159265358963 computed in 0.7989 seconds
running on 8 threads: PI = 3.141592653589769 computed in 0.7217 seconds
running on 9 threads: PI = 3.141592653589656 computed in 0.6415 seconds
running on 10 threads: PI = 3.141592653589794 computed in 0.5774 seconds
running on 11 threads: PI = 3.14159265358966 computed in 0.5249 seconds
running on 12 threads: PI = 3.14159265358986 computed in 0.4811 seconds
running on 13 threads: PI = 3.141592653589865 computed in 0.4441 seconds
running on 14 threads: PI = 3.141592653589788 computed in 0.4124 seconds
running on 15 threads: PI = 3.141592653589805 computed in 0.3849 seconds
running on 16 threads: PI = 3.141592653589832 computed in 0.3609 seconds
running on 17 threads: PI = 3.141592653589839 computed in 0.3397 seconds
running on 18 threads: PI = 3.141592653589814 computed in 0.3208 seconds
running on 19 threads: PI = 3.141592653589826 computed in 0.3053 seconds
running on 20 threads: PI = 3.141592653589855 computed in 0.2897 seconds
running on 21 threads: PI = 3.141592653589775 computed in 0.2768 seconds
running on 22 threads: PI = 3.141592653589823 computed in 0.2644 seconds
running on 23 threads: PI = 3.141592653589866 computed in 0.2528 seconds
running on 24 threads: PI = 3.141592653589792 computed in 0.2423 seconds
running on 25 threads: PI = 3.14159265358978 computed in 0.2326 seconds
running on 26 threads: PI = 3.141592653589832 computed in 0.2237 seconds
running on 27 threads: PI = 3.141592653589835 computed in 0.2154 seconds
running on 28 threads: PI = 3.141592653589816 computed in 0.2077 seconds
running on 29 threads: PI = 3.141592653589819 computed in 0.2006 seconds
running on 30 threads: PI = 3.141592653589893 computed in 0.1939 seconds
running on 31 threads: PI = 3.141592653589744 computed in 0.1876 seconds
running on 32 threads: PI = 3.141592653589758 computed in 0.1818 seconds
running on 33 threads: PI = 3.141592653589806 computed in 0.1763 seconds
running on 34 threads: PI = 3.141592653589926 computed in 0.1711 seconds
running on 35 threads: PI = 3.14159265358979 computed in 0.1662 seconds
running on 36 threads: PI = 3.141592653589822 computed in 0.1616 seconds
```

```
#!/bin/bash -l
#SBATCH --account=uzh8
#SBATCH --job-name=openmp_test
#SBATCH --time=01:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=36
#SBATCH --partition=normal
#SBATCH --constraint=mc

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

srun cpi_openmp
```

University of Zurich[UZH]

OpenMP Results (multi-threaded) N=1,000,000,000

running on 1 threads: PI = 3.141592653589971 computed in 4.55 seconds
running on 2 threads: PI = 3.141592653589901 computed in 2.291 seconds
running on 3 threads: PI = 3.141592653589962 computed in 1.624 seconds
running on 4 threads: PI = 3.141592653589821 computed in 1.258 seconds
running on 5 threads: PI = 3.141592653589596 computed in 1.041 seconds
running on 6 threads: PI = 3.141592653589682 computed in 0.8988 seconds
running on 7 threads: PI = 3.141592653589633 computed in 0.7989 seconds
running on 8 threads: PI = 3.141592653589769 computed in 0.7217 seconds
running on 9 threads: PI = 3.141592653589656 computed in 0.6415 seconds
running on 10 threads: PI = 3.141592653589794 computed in 0.5774 seconds
running on 11 threads: PI = 3.141592653589666 computed in 0.5249 seconds
running on 12 threads: PI = 3.14159265358986 computed in 0.4811 seconds
running on 13 threads: PI = 3.141592653589865 computed in 0.4441 seconds
running on 14 threads: PI = 3.141592653589788 computed in 0.4124 seconds
running on 15 threads: PI = 3.141592653589805 computed in 0.3849 seconds
running on 16 threads: PI = 3.141592653589832 computed in 0.3609 seconds
running on 17 threads: PI = 3.141592653589839 computed in 0.3397 seconds
running on 18 threads: PI = 3.141592653589814 computed in 0.3208 seconds
running on 19 threads: PI = 3.141592653589826 computed in 0.3053 seconds
running on 20 threads: PI = 3.141592653589855 computed in 0.2897 seconds
running on 21 threads: PI = 3.141592653589775 computed in 0.2768 seconds
running on 22 threads: PI = 3.141592653589823 computed in 0.2644 seconds
running on 23 threads: PI = 3.141592653589866 computed in 0.2528 seconds
running on 24 threads: PI = 3.141592653589792 computed in 0.2423 seconds
running on 25 threads: PI = 3.14159265358978 computed in 0.2326 seconds
running on 26 threads: PI = 3.141592653589832 computed in 0.2237 seconds
running on 27 threads: PI = 3.141592653589835 computed in 0.2154 seconds
running on 28 threads: PI = 3.141592653589816 computed in 0.2077 seconds
running on 29 threads: PI = 3.141592653589819 computed in 0.2006 seconds
running on 30 threads: PI = 3.141592653589893 computed in 0.1939 seconds
running on 31 threads: PI = 3.141592653589744 computed in 0.1876 seconds
running on 32 threads: PI = 3.141592653589758 computed in 0.1818 seconds
running on 33 threads: PI = 3.141592653589806 computed in 0.1763 seconds
running on 34 threads: PI = 3.141592653589926 computed in 0.1711 seconds
running on 35 threads: PI = 3.14159265358979 computed in 0.1662 seconds
running on 36 threads: PI = 3.141592653589822 computed in 0.1616 seconds

University of Zurich[UZH]

MPI Results

```
This is Process-1/36 running on nid00564
This is Process-3/36 running on nid00564
This is Process-5/36 running on nid00564
This is Process-6/36 running on nid00564
This is Process-8/36 running on nid00564
This is Process-11/36 running on nid00564
This is Process-13/36 running on nid00564
This is Process-14/36 running on nid00564
This is Process-17/36 running on nid00564
This is Process-18/36 running on nid00564
This is Process-22/36 running on nid00564
This is Process-23/36 running on nid00564
This is Process-25/36 running on nid00564
This is Process-26/36 running on nid00564
This is Process-28/36 running on nid00564
This is Process-29/36 running on nid00564
This is Process-31/36 running on nid00564
This is Process-33/36 running on nid00564
This is Process-35/36 running on nid00564
This is Process-0/36 running on nid00564
This is Process-2/36 running on nid00564
This is Process-4/36 running on nid00564
This is Process-7/36 running on nid00564
This is Process-9/36 running on nid00564
This is Process-10/36 running on nid00564
This is Process-12/36 running on nid00564
This is Process-15/36 running on nid00564
This is Process-16/36 running on nid00564
This is Process-19/36 running on nid00564
This is Process-20/36 running on nid00564
This is Process-21/36 running on nid00564
This is Process-24/36 running on nid00564
This is Process-27/36 running on nid00564
This is Process-30/36 running on nid00564
This is Process-32/36 running on nid00564
This is Process-34/36 running on nid00564
This program uses 36 processes
The number of intervals = 1000000000
```

pi is approximately 3.1415926535898406
Error is 0.0000000000000475
wall clock time = 0.096959

OpenMP Result
running on 36 threads: PI = 3.141592653589822 computed in 0.1616 seconds

Two Nodes

```
(some lines removed)
This is Process-62/72 running on nid01175
This is Process-22/72 running on nid01174
This is Process-63/72 running on nid01175
This is Process-23/72 running on nid01174
This is Process-64/72 running on nid01175
This is Process-24/72 running on nid01174
This is Process-65/72 running on nid01175
This is Process-25/72 running on nid01174
This is Process-66/72 running on nid01175
This is Process-26/72 running on nid01174
This is Process-67/72 running on nid01175
This is Process-29/72 running on nid01174
This is Process-68/72 running on nid01175
This is Process-30/72 running on nid01174
This is Process-69/72 running on nid01175
This is Process-31/72 running on nid01174
This is Process-70/72 running on nid01175
This is Process-32/72 running on nid01174
This is Process-71/72 running on nid01175
This is Process-33/72 running on nid01174
This is Process-39/72 running on nid01175
This is Process-34/72 running on nid01174
This is Process-40/72 running on nid01175
This is Process-35/72 running on nid01174
This is Process-41/72 running on nid01175
This is Process-0/72 running on nid01174
This is Process-43/72 running on nid01175
This is Process-5/72 running on nid01174
This is Process-44/72 running on nid01175
This is Process-7/72 running on nid01174
This is Process-45/72 running on nid01175
This is Process-11/72 running on nid01174
This is Process-46/72 running on nid01175
This is Process-13/72 running on nid01174
This is Process-47/72 running on nid01175
This is Process-19/72 running on nid01174
This is Process-51/72 running on nid01175
This is Process-20/72 running on nid01174
This is Process-59/72 running on nid01175
This is Process-27/72 running on nid01174
This is Process-28/72 running on nid01174
This program uses 72 processes
The number of intervals = 1000000000
```

pi is approximately 3.1415926535898109,
Error is 0.0000000000000178
wall clock time = 0.024506

```
#!/bin/bash -l
#SBATCH --account=uzh8
#SBATCH --job-name=openmp_test
#SBATCH --time=01:00:00
#SBATCH --nodes=(nodes)
#SBATCH --ntasks-per-node=(processes)
#SBATCH --cpus-per-task=(threads)
#SBATCH --partition=normal
#SBATCH --constraint=mc

srun cpi_mpi
```

University of Zurich[UZH]

MPI Results

```
This is Process-1/36 running on nid00564
This is Process-3/36 running on nid00564
This is Process-5/36 running on nid00564
This is Process-6/36 running on nid00564
This is Process-8/36 running on nid00564
This is Process-11/36 running on nid00564
This is Process-13/36 running on nid00564
This is Process-14/36 running on nid00564
This is Process-17/36 running on nid00564
This is Process-18/36 running on nid00564
This is Process-22/36 running on nid00564
This is Process-23/36 running on nid00564
This is Process-25/36 running on nid00564
This is Process-26/36 running on nid00564
This is Process-28/36 running on nid00564
This is Process-29/36 running on nid00564
This is Process-31/36 running on nid00564
This is Process-33/36 running on nid00564
This is Process-35/36 running on nid00564
This is Process-0/36 running on nid00564
This is Process-2/36 running on nid00564
This is Process-4/36 running on nid00564
This is Process-7/36 running on nid00564
This is Process-9/36 running on nid00564
This is Process-10/36 running on nid00564
This is Process-12/36 running on nid00564
This is Process-15/36 running on nid00564
This is Process-16/36 running on nid00564
This is Process-19/36 running on nid00564
This is Process-20/36 running on nid00564
This is Process-21/36 running on nid00564
This is Process-24/36 running on nid00564
This is Process-27/36 running on nid00564
This is Process-30/36 running on nid00564
This is Process-32/36 running on nid00564
This is Process-34/36 running on nid00564
This program uses 36 processes
The number of intervals = 1000000000
```

pi is approximately 3.1415926535898406
Error is 0.0000000000000475
wall clock time = 0.096959

OpenMP Result
running on 36 threads: PI = 3.141592653589822 computed in 0.1616 seconds

Two Nodes

```
(some lines removed)
This is Process-62/72 running on nid01175
This is Process-22/72 running on nid01174
This is Process-63/72 running on nid01175
This is Process-23/72 running on nid01174
This is Process-64/72 running on nid01175
This is Process-24/72 running on nid01174
This is Process-65/72 running on nid01175
This is Process-25/72 running on nid01174
This is Process-66/72 running on nid01175
This is Process-26/72 running on nid01174
This is Process-67/72 running on nid01175
This is Process-29/72 running on nid01174
This is Process-68/72 running on nid01175
This is Process-30/72 running on nid01174
This is Process-69/72 running on nid01175
This is Process-31/72 running on nid01174
This is Process-70/72 running on nid01175
This is Process-32/72 running on nid01174
This is Process-71/72 running on nid01175
This is Process-33/72 running on nid01174
This is Process-39/72 running on nid01175
This is Process-34/72 running on nid01174
This is Process-40/72 running on nid01175
This is Process-35/72 running on nid01174
This is Process-41/72 running on nid01175
This is Process-0/72 running on nid01174
This is Process-43/72 running on nid01175
This is Process-5/72 running on nid01174
This is Process-44/72 running on nid01175
This is Process-7/72 running on nid01174
This is Process-45/72 running on nid01175
This is Process-11/72 running on nid01174
This is Process-46/72 running on nid01175
This is Process-13/72 running on nid01174
This is Process-47/72 running on nid01175
This is Process-19/72 running on nid01174
This is Process-51/72 running on nid01175
This is Process-20/72 running on nid01174
This is Process-59/72 running on nid01175
This is Process-27/72 running on nid01174
This is Process-28/72 running on nid01174
This program uses 72 processes
The number of intervals = 1000000000
```

```
#!/bin/bash -l
#SBATCH --account=uzh8
#SBATCH --job-name=openmp_test
#SBATCH --time=01:00:00
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=36
#SBATCH --cpus-per-task=1
#SBATCH --partition=normal
#SBATCH --constraint=mc

srun cpi_mpi
```

pi is approximately 3.1415926535898109,
Error is 0.0000000000000178
wall clock time = 0.024506

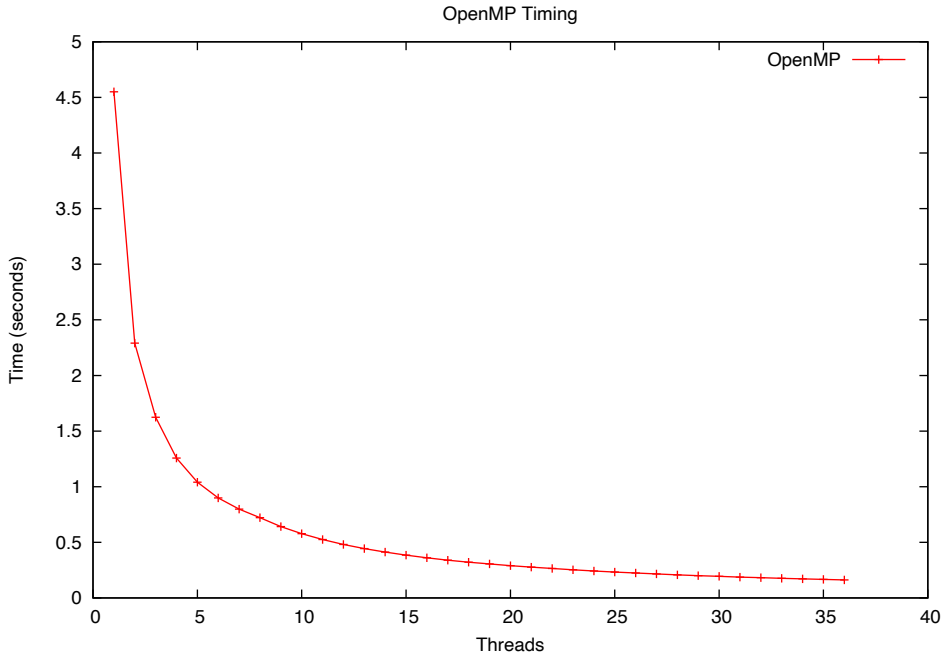
Last week

Same results in table format

THREADS	SECONDS	THREADS	SECONDS
1	4.550	19	0.3053
2	2.291	20	0.2897
3	1.624	21	0.2768
4	1.258	22	0.2644
5	1.041	23	0.2528
6	0.8988	24	0.2423
7	0.7989	25	0.2326
8	0.7217	26	0.2237
9	0.6415	27	0.2154
10	0.5774	28	0.2077
11	0.5249	29	0.2006
12	0.4811	30	0.1939
13	0.4441	31	0.1876
14	0.4124	32	0.1818
15	0.3849	33	0.1763
16	0.3609	34	0.1711
17	0.3397	35	0.1662
18	0.3208	36	0.1616

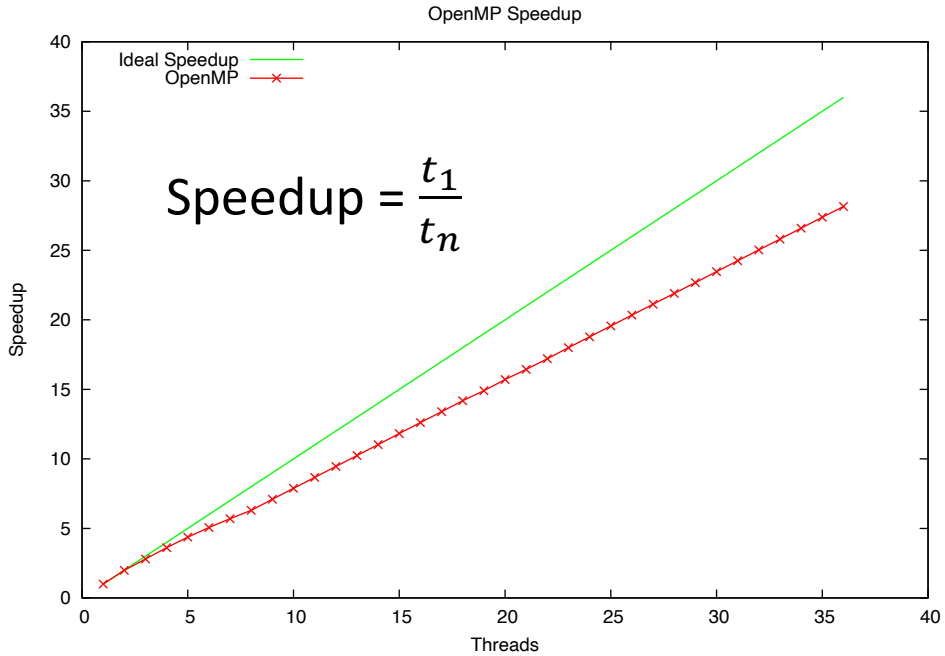
Timing Plot

```
set title 'OpenMP Timing'
set xlabel 'Threads'
set ylabel 'Time (seconds)'
set key top right
plot "cpi_openmp.dat”\
 u 1:2 w lp lw 2 t 'OpenMP'
```



“Speedup” Plot

```
set title 'OpenMP Speedup'
set xlabel 'Threads'
set ylabel 'Speedup'
set key top left
plot x lc 2 lw 2 t 'Ideal Speedup’,\
"cpi_openmp.dat" u 1:(4.55/$2)\
 w lp lc 1 lw 2 t 'OpenMP'
```



University of Zurich[UZH]

Parsing the Text (Results)

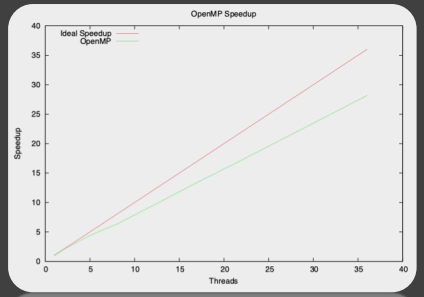
running on 1 threads: PI = 3.141592653589971 computed in 4.55 seconds

1	**4.550000**	13	0.444100	25	0.232600
2	2.291000	14	0.412400	26	0.223700
3	1.624000	15	0.384900	27	0.215400
4	1.258000	16	0.360900	28	0.207700
5	1.041000	17	0.339700	29	0.200600
6	0.898800	18	0.320800	30	0.193900
7	0.798900	19	0.305300	31	0.187600
8	0.721700	20	0.289700	32	0.181800
9	0.641500	21	0.276800	33	0.176300
10	0.577400	22	0.264400	34	0.171100
11	0.524900	23	0.252800	35	0.166200
12	0.481100	24	0.242300	36	0.161600

./parse.pl <slurm-6188431.out

University of Zurich[UZH]

How to Plot the Results




```
dpotter@daint104:~/hpc1a> cat parse.pl
#!/usr/bin/perl
use strict;

while(<STDIN>) {
    if (/running on (\d+) threads: PI.*computed in ([0-9]*\.[0-9]+) seconds/) {
          printf("%2d %f\n",$1,$2);
          }
    }
```

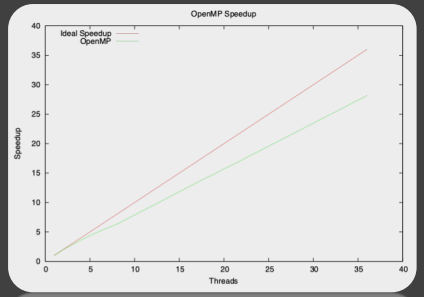
./parse.pl <slurm-6188431.out >cpi_openmp.dat

```
dpotter@daint104:~/hpc1a> cat speedup.gp
set title 'OpenMP Speedup'
set xlabel 'Threads'
set ylabel 'Speedup'
set key top left
plot x t 'Ideal Speedup', "cpi_openmp.dat" u 1:(4.55/$2) w l t 'OpenMP'
```

```
gnuplot> set output "speedup.eps"
gnuplot> set terminal postscript enhanced solid color
gnuplot> load "speedup.gp"
gnuplot> set terminal x11
gnuplot> set output
```

University of Zurich[UZH]

Parsing with Python




```
dpotter@daint104:~/hpc1a> cat parse.py
#!/usr/bin/env python
import re, sys

for line in re.finditer(
  r"running on ([0-9]+) threads: PI.*computed in ([0-9]*\.[0-9]+) seconds",
  sys.stdin.read(), re.MULTILINE):
  print("{} {}".format(line.group(1),line.group(2)))
```

./parse.py <slurm-6188431.out >cpi_openmp.dat

```
dpotter@daint104:~/hpc1a> cat speedup.gp
set title 'OpenMP Speedup'
set xlabel 'Threads'
set ylabel 'Speedup'
set key top left
plot x t 'Ideal Speedup', "cpi_openmp.dat" u 1:(4.55/$2) w l t 'OpenMP'
```

```
gnuplot> set output "speedup.eps"
gnuplot> set terminal postscript enhanced solid color
gnuplot> load "speedup.gp"
gnuplot> set terminal x11
gnuplot> set output
```

Parsing Process

```
[dpotter@ela4 hpc1a]$ head -n 36 slurm-6188431.out
running on 1 threads: PI = 3.141592653589971 computed in 4.55 seconds
running on 2 threads: PI = 3.141592653589901 computed in 2.291 seconds
running on 3 threads: PI = 3.141592653589962 computed in 1.624 seconds
running on 4 threads: PI = 3.141592653589821 computed in 1.258 seconds
running on 5 threads: PI = 3.141592653589596 computed in 1.041 seconds
running on 6 threads: PI = 3.141592653589682 computed in 0.8988 seconds
running on 7 threads: PI = 3.14159265358963 computed in 0.7989 seconds
running on 8 threads: PI = 3.141592653589769 computed in 0.7217 seconds
running on 9 threads: PI = 3.141592653589656 computed in 0.6415 seconds
running on 10 threads: PI = 3.141592653589794 computed in 0.5774 seconds
running on 11 threads: PI = 3.14159265358966 computed in 0.5249 seconds
running on 12 threads: PI = 3.14159265358986 computed in 0.4811 seconds
running on 13 threads: PI = 3.141592653589865 computed in 0.4441 seconds
running on 14 threads: PI = 3.141592653589788 computed in 0.4124 seconds
running on 15 threads: PI = 3.141592653589805 computed in 0.3849 seconds
running on 16 threads: PI = 3.141592653589832 computed in 0.3609 seconds
running on 17 threads: PI = 3.141592653589839 computed in 0.3397 seconds
running on 18 threads: PI = 3.141592653589814 computed in 0.3208 seconds
running on 19 threads: PI = 3.141592653589826 computed in 0.3053 seconds
running on 20 threads: PI = 3.141592653589855 computed in 0.2897 seconds
running on 21 threads: PI = 3.141592653589775 computed in 0.2768 seconds
running on 22 threads: PI = 3.141592653589823 computed in 0.2644 seconds
running on 23 threads: PI = 3.141592653589866 computed in 0.2528 seconds
running on 24 threads: PI = 3.141592653589792 computed in 0.2423 seconds
running on 25 threads: PI = 3.14159265358978 computed in 0.2326 seconds
running on 26 threads: PI = 3.141592653589832 computed in 0.2237 seconds
running on 27 threads: PI = 3.141592653589835 computed in 0.2154 seconds
running on 28 threads: PI = 3.141592653589816 computed in 0.2077 seconds
running on 29 threads: PI = 3.141592653589819 computed in 0.2006 seconds
running on 30 threads: PI = 3.141592653589893 computed in 0.1939 seconds
running on 31 threads: PI = 3.141592653589744 computed in 0.1876 seconds
running on 32 threads: PI = 3.141592653589758 computed in 0.1818 seconds
running on 33 threads: PI = 3.141592653589806 computed in 0.1763 seconds
running on 34 threads: PI = 3.141592653589926 computed in 0.1711 seconds
running on 35 threads: PI = 3.14159265358979 computed in 0.1662 seconds
running on 36 threads: PI = 3.141592653589822 computed in 0.1616 seconds
```

```
[dpotter@ela4 hpc1a]$ python parse.py <slurm-6188431.out
1 4.55
2 2.291
3 1.624
4 1.258
5 1.041
6 0.8988
7 0.7989
8 0.7217
9 0.6415
10 0.5774
11 0.5249
12 0.4811
13 0.4441
14 0.4124
15 0.3849
16 0.3609
17 0.3397
18 0.3208
19 0.3053
20 0.2897
21 0.2768
22 0.2644
23 0.2528
24 0.2423
25 0.2326
26 0.2237
27 0.2154
28 0.2077
29 0.2006
30 0.1939
31 0.1876
32 0.1818
33 0.1763
34 0.1711
35 0.1662
36 0.1616
```

More on Compilers

Compiler Suites

- Cray Compiler (PrgEnv-cray)
- GNU Compiler (PrgEnv-gnu)
  - Free for everyone
- Intel Compiler (PrgEnv-intel)
  - Free for Students
- Portland Compiler (PrgEnv-pgi)
  - Important for OpenACC (GPU)
- clang (Apple mostly)
  - Open Source & GNU Compatible
- Visual Studio (Windows)
  - Free for us through Microsoft Imagine

```
#include <stdio.h>
#include <sys/time.h>

static long steps = 1000000000;

double getTime(void) {
    struct timeval tv;
    struct timezone tz;
    gettimeofday(&tv, &tz);
    return tv.tv_sec + 1e-6*(double)tv.tv_usec;
}

int main (int argc, const char *argv[]) {
    int i;
    double x;
    double pi;
    char *p;

    double step = 1.0/(double) steps;
    double sum = 0.0;
    double start = getTime();

    for (i=0; i < steps; i++) {
        x = (i+0.5)*step;
        sum += 4.0 / (1.0+x*x);
    }
    pi = step * sum;
    double delta = getTime() - start;
    printf("PI = %.16g computed in %.4g seconds\n", pi, delta);
}
```

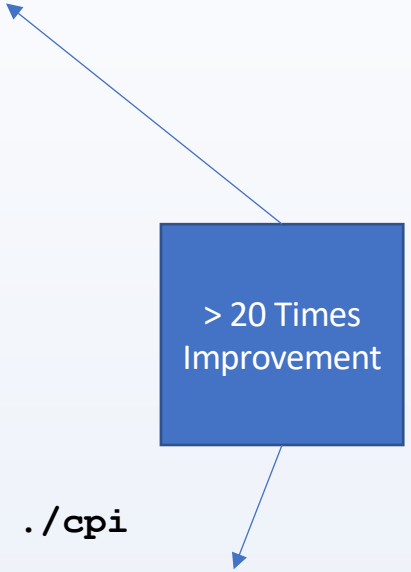
cpi.c (serial version)

Output Name

```
dhcp-94-191:cpi$ ls
cpi.c
dhcp-94-191:cpi$ gcc cpi.c
dhcp-94-191:cpi$ ls
a.out cpi.c
dhcp-94-191:cpi$ ./a.out
PI = 3.141592653589971 computed in 11.9 seconds
dhcp-94-191:cpi$ rm a.out
dhcp-94-191:cpi$ ls
cpi.c
dhcp-94-191:cpi$ gcc -o cpi cpi.c
dhcp-94-191:cpi$ ls
cpi   cpi.c
dhcp-94-191:cpi$ ./cpi
PI = 3.141592653589971 computed in 11.82 seconds
```

Compiler Optimization

```
dhcp-94-191:cpi$ gcc -o cpi cpi.c ; ./cpi
PI = 3.141592653589971 computed in 11.84 seconds
dhcp-94-191:cpi$ gcc -O -o cpi cpi.c ; ./cpi
PI = 3.141592653589971 computed in 2.505 seconds
dhcp-94-191:cpi$ gcc -O0 -o cpi cpi.c ; ./cpi
PI = 3.141592653589971 computed in 11.82 seconds
dhcp-94-191:cpi$ gcc -O1 -o cpi cpi.c ; ./cpi
PI = 3.141592653589971 computed in 2.497 seconds
dhcp-94-191:cpi$ gcc -O2 -o cpi cpi.c ; ./cpi
PI = 3.141592653589971 computed in 1.19 seconds
dhcp-94-191:cpi$ gcc -O3 -o cpi cpi.c ; ./cpi
PI = 3.141592653589971 computed in 1.174 seconds
dhcp-94-191:cpi$ gcc -O3 -ffast-math -o cpi cpi.c ; ./cpi
PI = 3.141592653589652 computed in 0.6 seconds
dhcp-94-191:cpi$ gcc -O3 -ffast-math -mavx2 -o cpi cpi.c ; ./cpi
PI = 3.141592653589729 computed in 0.5518 seconds
     3.14159265358979323846 (real value of pi)
```



What do the levels mean?

-O1 Optimize.

- Optimizing compilation takes somewhat more time, and a lot more memory for a large function.

-O2 Optimize even more.

- GCC performs nearly all supported optimizations that do not involve a space-speed tradeoff.
- The compiler does not perform loop unrolling or function inlining when you specify -O2.
- As compared to -O, this option **increases both compilation time** and the performance of the generated code.

-O3 Optimize yet more.

- -O3 turns on all optimizations specified by -O2 and also turns on the -finline-functions, -funswitch-loops, -fpredictive-commoning, -fgcse-after-reload and -ftree-vectorize options.

-O0 No Optimization.

- Reduce compilation time and make debugging produce the expected results.
- **This is the default**.

Fast Math

Associative Property of Addition

$$(a + b) + c = a + (b + c)$$

Distributive Property of Multiplication

$$a(b + c) = ab + ac$$

Sometimes reordering can improve performance

Fast Math (the Truth)

Associative Property of Addition

$$(a + b) + c \neq a + (b + c)$$

Distributive Property of Multiplication

$$a(b + c) \neq ab + ac$$

Sometimes reordering can change the result

Beware of Associative Math

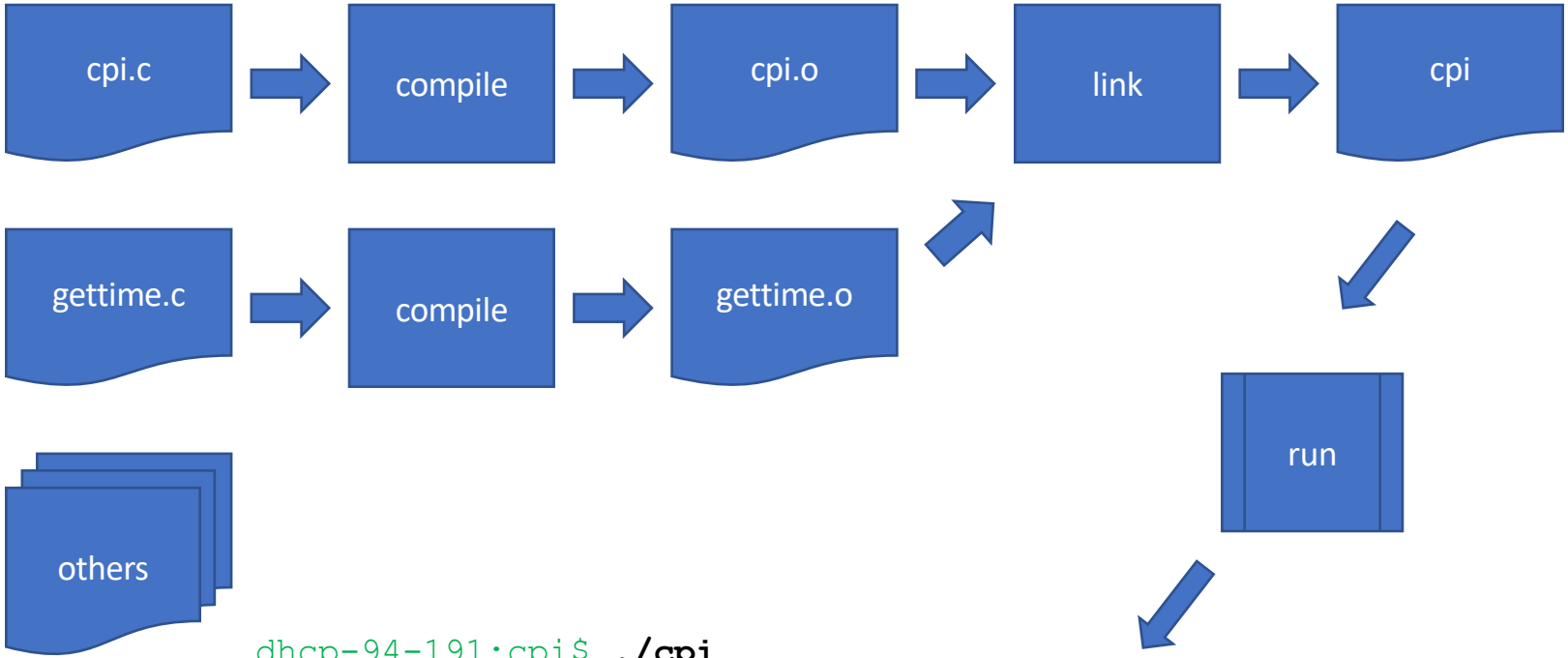
```
dhcp-94-191:cpi$ python
>>> a=12
>>> b=12
>>> a==b
True
>>> a=0.1 + 0.2 + 0.3
>>> b=0.1 +(0.2 + 0.3)
>>> a==b
False
>>> print(a)
0.6
>>> print(b)
0.6
```

```
>>> print("%.17f" %a)
0.60000000000000009
>>> print("%.17f" %b)
0.59999999999999998
>>> c=0.6
>>> print("%.17f" %c)
0.59999999999999998
```

“Modules” in C

```
#include <stdio.h>
#include <sys/time.h>

static long steps = 1000000000;

double getTime(void) {
    struct timeval tv;
    struct timezone tz;
    gettimeofday(&tv, &tz);
    return tv.tv_sec + 1e-6*(double)tv.tv_usec;
}

int main (int argc, const char *argv[]) {
    int i;
    double x;
    double pi;

    double step = 1.0/(double) steps;
    double sum = 0.0;
    double start = getTime();

    for (i=0; i < steps; i++) {
        x = (i+0.5)*step;
        sum += 4.0 / (1.0+x*x);
    }
    pi = step * sum;
    double delta = getTime() - start;
    printf("PI = %.16g computed in %.4g seconds\n", pi, delta);
}
```

Separate Compilation




```
dhcp-94-191:cpi$ ./cpi
PI = 3.141592653589971 computed in 1.174 seconds
```

gettime.c

```
#include <sys/time.h>

double getTime(void) {
    struct timeval tv;
    struct timezone tz;
    gettimeofday(&tv, &tz);
    return tv.tv_sec + 1e-6*(double)tv.tv_usec;
}
```

cpi.c

```
#include <stdio.h>

static long steps = 1000000000;

int main (int argc, const char *argv[]) {
    int i;
    double x;
    double pi;

    double step = 1.0/(double) steps;
    double sum = 0.0;
    double start = getTime();

    for (i=0; i < steps; i++) {
        x = (i+0.5)*step;
        sum += 4.0 / (1.0+x*x);
    }
    pi = step * sum;
    double delta = getTime() - start;
    printf("PI = %.16g computed in %.4g seconds\n", pi, delta);
}
```

Compiling with two files

```
dhcp-94-191:cpi2$ gcc -O3 -ffast-math -mavx2 -o cpi cpi.c gettime.c
cpi.c: In function 'main':
cpi.c:13:20: warning: implicit declaration of function 'getTime'; did you mean 'getline'? [-Wimplicit-function-declaration]
     double start = getTime();
                    ^~~~~~~
                    getline
```

gettime.h

```
double getTime(void);
```

cpi.c

```
#include <stdio.h>
#include "gettime.h"

static long steps = 1000000000;

int main (int argc, const char *argv[]) {
    int i;
    double x;
    double pi;

    double step = 1.0/(double) steps;
    double sum = 0.0;
    double start = getTime();
    for (i=0; i < steps; i++) {
        x = (i+0.5)*step;
        sum += 4.0 / (1.0+x*x);
    }
    pi = step * sum;
    double delta = getTime() - start;
    printf("PI = %.16g computed in %.4g seconds\n", pi, delta);
}
```

gettime.h

```
double getTime(void);
```

gettime.c

```
#include <sys/time.h>
#include "gettime.h"

double getTime(void) {
    struct timeval tv;
    struct timezone tz;
    gettimeofday(&tv, &tz);
    return tv.tv_sec + 1e-6*(double)tv.tv_usec;
}
```

gettime.h

```
double getTime(void);
```

Compile and Link Separately

```
dhcp-94-191:cpi2$ gcc -O3 -ffast-math -mavx2 -o cpi cpi.c gettime.c
dhcp-94-191:cpi2$ ls
cpi       cpi.c     gettime.c gettime.h
dhcp-94-191:cpi2$ rm cpi


dhcp-94-191:cpi2$ ls
cpi.c     gettime.c gettime.h
dhcp-94-191:cpi2$ gcc -O3 -ffast-math -mavx2 -c -o cpi.o cpi.c
dhcp-94-191:cpi2$ ls
cpi.c     cpi.o     gettime.c gettime.h
dhcp-94-191:cpi2$ gcc -O3 -ffast-math -mavx2 -c -o gettime.o gettime.c
dhcp-94-191:cpi2$ ls
cpi.c     cpi.o     gettime.c gettime.h gettime.o
dhcp-94-191:cpi2$ gcc -o cpi cpi.o gettime.o
dhcp-94-191:cpi2$ ./cpi
PI = 3.141592653589729 computed in 0.567 seconds
```

Makefile

```
cpi             : cpi.o gettime.o
        gcc -o cpi cpi.o gettime.o

cpi.o           : cpi.c gettime.h
        gcc -O3 -ffast-math -mavx2 -c -o cpi.o cpi.c

gettime.o       : gettime.c gettime.h
        gcc -O3 -ffast-math -mavx2 -c -o gettime.o gettime.c

clean:
        rm -f cpi cpi.o gettime.o
```

Compile and Link Separately

```
dhcp-94-191:cpi2$ make clean
rm -f cpi cpi.o gettime.o
dhcp-94-191:cpi2$ make
gcc -O3 -ffast-math -mavx2 -c -o cpi.o cpi.c
gcc -O3 -ffast-math -mavx2 -c -o gettime.o gettime.c
gcc -o cpi cpi.o gettime.o
dhcp-94-191:cpi2$ make
make: `cpi' is up to date.
dhcp-94-191:cpi2$ touch cpi.c
dhcp-94-191:cpi2$ make
gcc -O3 -ffast-math -mavx2 -c -o cpi.o cpi.c
gcc -o cpi cpi.o gettime.o
```

"touch" – change modification time.
Same as if you edited the file.

Makefile

```
cpi       : cpi.o gettime.o
        gcc -o cpi cpi.o gettime.o

cpi.o     : cpi.c gettime.h
        gcc -O3 -ffast-math -mavx2 -c -o cpi.o cpi.c

gettime.o: gettime.c gettime.h
        gcc -O3 -ffast-math -mavx2 -c -o gettime.o gettime.c

clean:
        rm -f cpi cpi.o gettime.o
```

Makefile with default rules

```
cpi       : cpi.o gettime.o

cpi.o     : cpi.c gettime.h

gettime.o: gettime.c gettime.h

clean:
   rm -f cpi cpi.o gettime.o
```

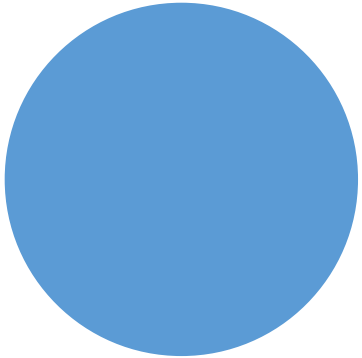
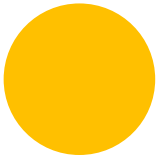
```
dhcp-94-191:cpi2$ make clean
rm -f cpi cpi.o gettime.o
dhcp-94-191:cpi2$ make
cc    -c -o cpi.o cpi.c
cc    -c -o gettime.o gettime.c
cc   cpi.o gettime.o   -o cpi
```

University of Zurich[UZH]

Makefile with default rules

```
CFLAGS=-O3 -ffast-math -mavx2
CC=gcc

cpi       : cpi.o gettime.o

cpi.o     : cpi.c gettime.h

gettime.o: gettime.c gettime.h

clean:
   rm -f cpi cpi.o gettime.o
```

```
dhcp-94-191:cpi2$ make clean
rm -f cpi cpi.o gettime.o
dhcp-94-191:cpi2$ make
gcc -O3 -ffast-math -mavx2     -c -o cpi.o cpi.c
gcc -O3 -ffast-math -mavx2     -c -o gettime.o gettime.c
gcc    cpi.o gettime.o    -o cpi
```

Message Passing

Why use MPI?

One Node isn't enough

- Maybe you have run out of memory
- Maybe it takes too long to calculate

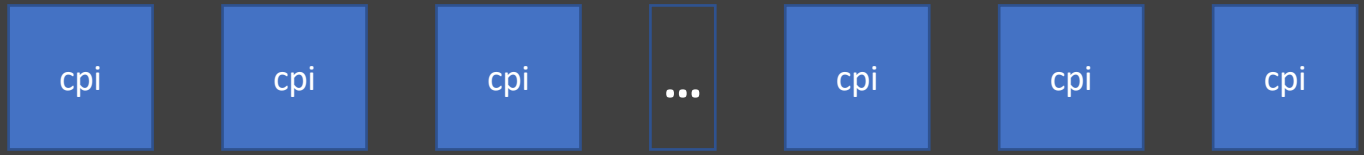
Alternatives do exist

- Partitioned Global Address Space (PGAS)
- HPX (High Performance ParalleX)
- Charm++

MPI is still omnipresent

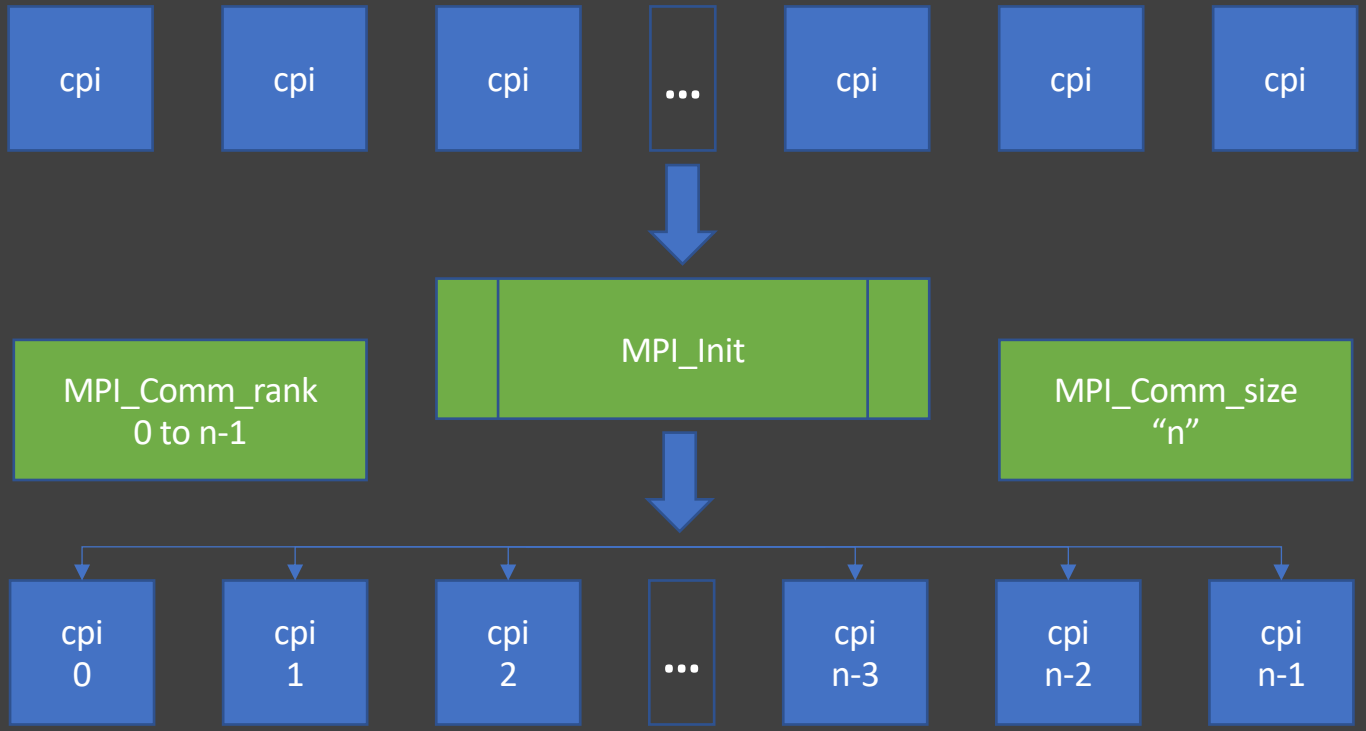
- It has been around since 1991
- Version 3.1 was approved in 2015
- Version 4.0 release candidate now (2021)

MPI Process Layout

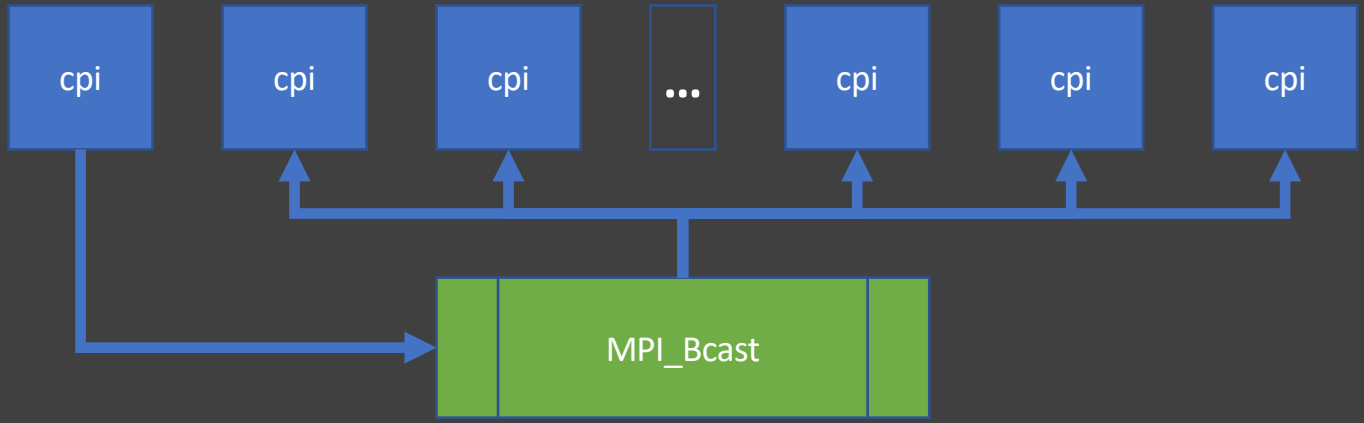


University of Zurich[UZH]

MPI Initialization

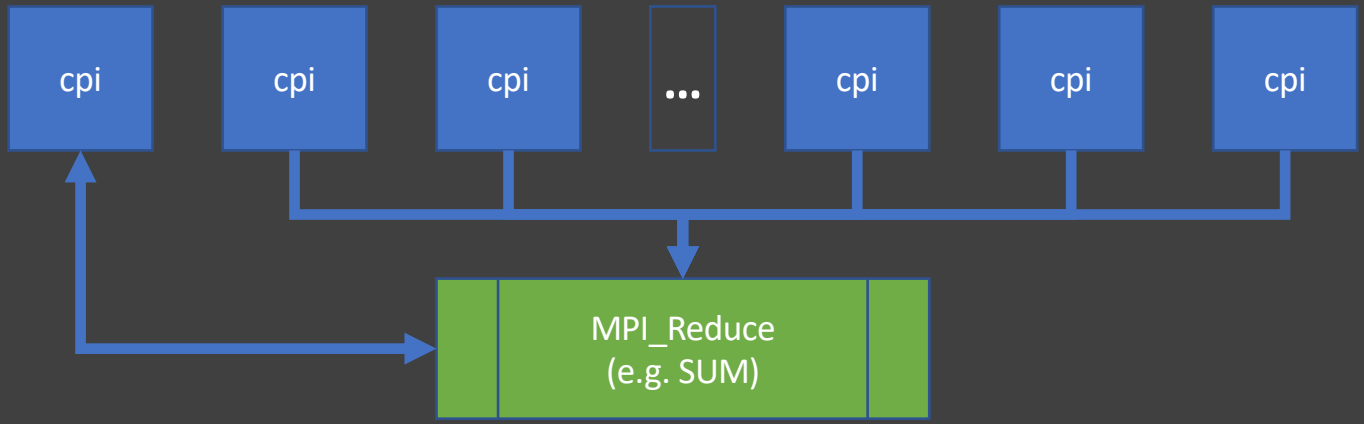


MPI Broadcast



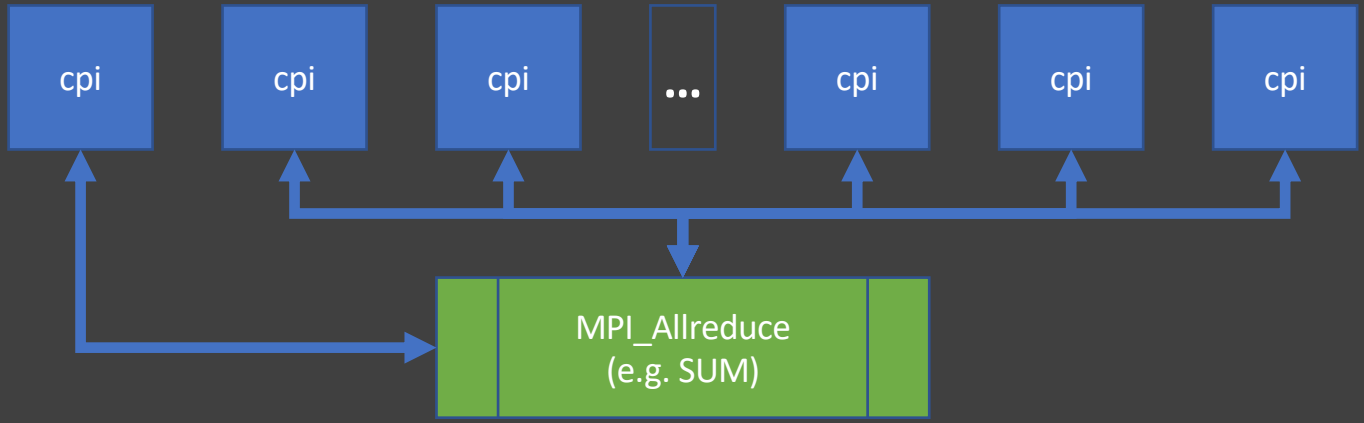
All MPI Ranks Call MPI_Bcast together.
Everyone gets the value from (usually) rank 0.

MPI Reduce



All MPI Ranks Call MPI_Reduce together.
All values from all ranks are summed together.
A single rank (usually rank 0) gets the result.

MPI All Reduce

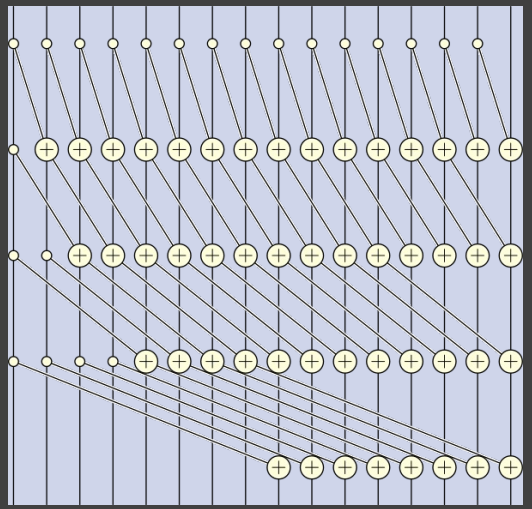


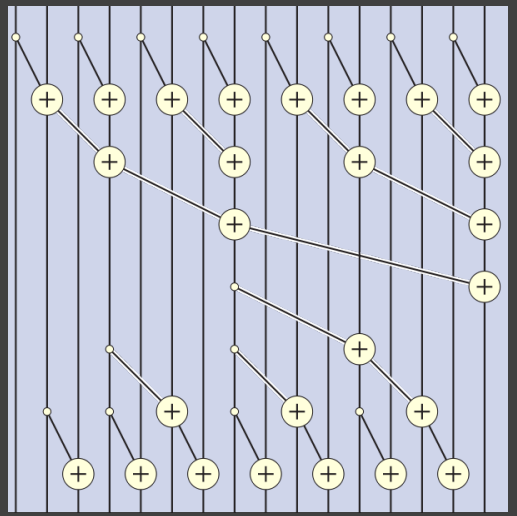
All MPI Ranks Call MPI_Reduce together.
All values from all ranks are summed together.
All ranks get the result.

University of Zurich[UZH]

Partial Reduction (e.g., sum)

Rank	0	1	2	3	4	5	6	7
Value	4	3	8	7	2	6	3	1
Scan	4	7	15	22	24	30	33	34
Exscan	0	4	7	15	22	24	30	33



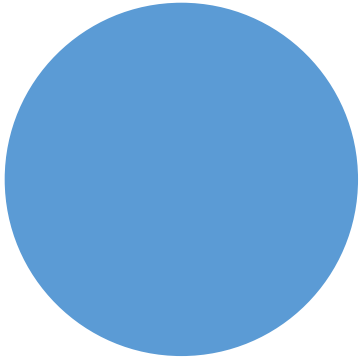
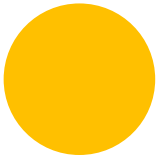


The MPI version of cpi

```
MPI_Init(&argc,&argv);                    /* Connect processes to each other */
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);   /* Get total number of processes */
MPI_Comm_rank(MPI_COMM_WORLD,&myid);              /* Rank of this process */
MPI_Get_processor_name(name, &resultlen);

printf("This is Process-%d/%d running on %s \n",myid,numprocs,name);
MPI_Barrier(MPI_COMM_WORLD);

if(myid == 0) {
    printf("This program uses %d processes\n", numprocs);
    n = 1000000000;
    }

MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);/* Send n from 0 to all others */

sum = 0.0;
h   = 1.0/n;
for (i=myid+0.5; i<n; i+=numprocs) {
    sum += dx_arctan(i*h);
    }
mypi = 4.0*h*sum;

                  /* Add all partial sums to each other and send to rank 0 */
MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

if (myid == 0) {
    printf("pi is approximately %.16f\n",pi);
    }
MPI_Finalize();                                /* Disconnect all processes */
```

```
/* Compute the Derivative of ArcTan */
double dx_arctan(double x) {
    return 1.0 / (1.0 + x*x);
}
```

MPI_Comm_accept
MPI_Comm_c2f
MPI_Comm_call_errhandler
MPI_Comm_compare
MPI_Comm_connect
MPI_Comm_create
MPI_Comm_create_errhandler
MPI_Comm_create_group
MPI_Comm_create_keyval
MPI_Comm_delete_attr
MPI_Comm_disconnect
MPI_Comm_dup
MPI_Comm_dup_with_info
MPI_Comm_f2c
MPI_Comm_free
MPI_Comm_free_keyval
MPI_Comm_get_attr
MPI_Comm_get_errhandler
MPI_Comm_get_info
MPI_Comm_get_name
MPI_Comm_get_parent
MPI_Comm_group
MPI_Comm_idup
MPI_Comm_join
MPI_Comm_rank
MPI_Comm_remote_group
MPI_Comm_remote_size
MPI_Comm_set_attr
MPI_Comm_set_errhandler
MPI_Comm_set_info
MPI_Comm_set_name
MPI_Comm_size
MPI_Comm_spawn
MPI_Comm_spawn_multiple
MPI_Comm_split
MPI_Comm_split_type
MPI_Comm_test_inter
MPI_Compare_and_swap
MPI_Dims_create
MPI_Dist_graph_create
MPI_Dist_graph_create_adjacent
MPI_Dist_graph_neighbors
MPI_Dist_graph_neighbors_count
MPI_File_write_ordered_end
MPI_File_write_shared
MPI_Finalize
MPI_Finalized
MPI_Free_mem
MPI_Gather
MPI_Gatherv
MPI_Get
MPI_Get_accumulate
MPI_Get_address
MPI_Get_count
MPI_Get_elements
MPI_Get_elements_x
MPI_Get_library_version
MPI_Get_processor_name
MPI_Get_version
MPI_Graph_create
MPI_Graph_get
MPI_Graph_map
MPI_Graph_neighbors
MPI_Graph_neighbors_count
MPI_Graphdims_get
MPI_Grequest_complete
MPI_Grequest_start
MPI_Group_c2f
MPI_Group_compare
MPI_Group_difference
MPI_Group_excl
MPI_Group_f2c
MPI_Group_free
MPI_Group_incl
MPI_Group_intersection
MPI_Group_range_excl
MPI_Group_range_incl
MPI_Group_rank
MPI_Group_size
MPI_Group_translate_ranks
MPI_Group_union
MPI_Iallgather
MPI_Iallgatherv
MPI_Iallreduce
MPI_Ialltoall
MPI_Ialltoallv
MPI_Publish_name
MPI_Put
MPI_Query_thread
MPI_Raccumulate
MPI_Recv
MPI_Recv_init
MPI_Reduce
MPI_Reduce_local
MPI_Reduce_scatter
MPI_Reduce_scatter_block
MPI_Register_datarep
MPI_Request_c2f
MPI_Request_f2c
MPI_Request_free
MPI_Request_get_status
MPI_Rget
MPI_Rget_accumulate
MPI_Rput
MPI_Rsend
MPI_Rsend_init
MPI_Scan
MPI_Scatter
MPI_Scatterv
MPI_Send
MPI_Send_init
MPI_Sendrecv
MPI_Sendrecv_replace
MPI_Sizeof
MPI_Ssend
MPI_Ssend_init
MPI_Start
MPI_Startall
MPI_Status_c2f
MPI_Status_f2c
MPI_Status_set_cancelled
MPI_Status_set_elements
MPI_Status_set_elements_x
MPI_T_category_changed
MPI_T_category_get_categories
MPI_T_category_get_cvars
MPI_T_category_get_info
MPI_T_category_get_num
MPI_T_category_get_pvars
MPI_File_call_errhandler
MPI_File_close
MPI_File_create_errhandler
MPI_File_delete
MPI_File_f2c
MPI_File_get_amode
MPI_File_get_atomicity
MPI_File_get_byte_offset
MPI_File_get_errhandler
MPI_File_get_group
MPI_File_get_info
MPI_File_get_position
MPI_File_get_position_shared
MPI_File_get_size
MPI_File_get_type_extent
MPI_File_get_view
MPI_File_iread
MPI_File_iread_all
MPI_File_iread_at
MPI_File_iread_at_all
MPI_File_iread_shared
MPI_File_iwrite
MPI_File_iwrite_all
MPI_File_iwrite_at
MPI_File_iwrite_at_all
MPI_File_iwrite_shared
MPI_File_open
MPI_File_preallocate
MPI_File_read
MPI_File_read_all
MPI_File_read_all_begin
MPI_File_read_all_end
MPI_File_read_at
MPI_File_read_at_all
MPI_File_read_at_all_begin
MPI_File_read_at_all_end
MPI_File_read_ordered
MPI_File_read_ordered_begin
MPI_File_read_ordered_end
MPI_File_read_shared
MPI_File_seek
MPI_File_seek_shared
MPI_File_set_atomicity
MPI_File_set_errhandler
MPI_File_set_info
MPI_File_set_size
MPI_File_set_view
MPI_File_sync
MPI_File_write
MPI_File_write_all
MPI_File_write_all_begin
MPI_File_write_all_end
MPI_File_write_at
MPI_File_write_at_all
MPI_File_write_at_all_begin
MPI_File_write_at_all_end
MPI_File_write_ordered
MPI_File_write_ordered_begin
MPI_File_write_ordered_end
MPI_File_write_shared
MPI_Finalize
MPI_Finalized
MPI_Free_mem
MPI_Gather
MPI_Gatherv
MPI_Get
MPI_Get_accumulate
MPI_Get_address
MPI_Get_count
MPI_Get_elements
MPI_Get_elements_x
MPI_Get_library_version
MPI_Get_processor_name
MPI_Get_version
MPI_Graph_create
MPI_Graph_get
MPI_Graph_map
MPI_Graph_neighbors
MPI_Graph_neighbors_count
MPI_Graphdims_get
MPI_Grequest_complete
MPI_Grequest_start
MPI_Group_c2f
MPI_Group_compare
MPI_Group_difference
MPI_Group_excl
MPI_Group_f2c
MPI_Group_free
MPI_Group_incl
MPI_Group_intersection
MPI_Group_range_excl
MPI_Group_range_incl
MPI_Group_rank
MPI_Group_size
MPI_Group_translate_ranks
MPI_Group_union
MPI_Iallgather
MPI_Iallgatherv
MPI_Iallreduce
MPI_Ialltoall
MPI_Ialltoallv
MPI_Ineighbor_allgather
MPI_Ineighbor_allgatherv
MPI_Ineighbor_alltoall
MPI_Ineighbor_alltoallv
MPI_Ineighbor_alltoallw
MPI_Info_c2f
MPI_Info_create
MPI_Info_delete
MPI_Info_dup
MPI_Info_env
MPI_Info_f2c
MPI_Info_free
MPI_Info_get
MPI_Info_get_nkeys
MPI_Info_get_nthkey
MPI_Info_get_valuelen
MPI_Info_set
MPI_Init
MPI_Init_thread
MPI_Initialized
MPI_Intercomm_create
MPI_Intercomm_merge
MPI_Iprobe
MPI_Irecv
MPI_Ireduce
MPI_Ireduce_scatter
MPI_Ireduce_scatter_block
MPI_Irsend
MPI_Is_thread_main
MPI_Iscan
MPI_Iscatter
MPI_Iscatterv
MPI_Isend
MPI_Issend
MPI_Keyval_create
MPI_Keyval_free
MPI_Lookup_name
MPI_Message_c2f
MPI_Message_f2c
MPI_Mprobe
MPI_Mrecv
MPI_Neighbor_allgather
MPI_Neighbor_allgatherv
MPI_Neighbor_alltoall
MPI_Neighbor_alltoallv
MPI_Neighbor_alltoallw
MPI_Op_c2f
MPI_Op_commutative
MPI_Op_create
MPI_Op_f2c
MPI_Op_free
MPI_Open_port
MPI_Pack
MPI_Pack_external
MPI_T_init_thread
MPI_T_pvar_get_info
MPI_T_pvar_get_num
MPI_T_pvar_handle_alloc
MPI_T_pvar_handle_free
MPI_T_pvar_read
MPI_T_pvar_readreset
MPI_T_pvar_reset
MPI_T_pvar_session_create
MPI_T_pvar_session_free
MPI_T_pvar_start
MPI_T_pvar_stop
MPI_T_pvar_write
MPI_Test
MPI_Test_cancelled
MPI_Testall
MPI_Testany
MPI_Testsome
MPI_Topo_test
MPI_Type_c2f
MPI_Type_commit
MPI_Type_contiguous
MPI_Type_create_darray
MPI_Type_create_f90_complex
MPI_Type_create_f90_integer
MPI_Type_create_f90_real
MPI_Type_create_hindexed
MPI_Type_create_hindexed_block
MPI_Type_create_hvector
MPI_Type_create_indexed_block
MPI_Type_create_keyval
MPI_Type_create_resized
MPI_Type_create_struct
MPI_Type_create_subarray
MPI_Type_delete_attr
MPI_Type_dup
MPI_Type_extent
MPI_Type_f2c
MPI_Type_free
MPI_Type_free_keyval
MPI_Type_get_attr
MPI_Type_get_contents
MPI_Type_get_envelope
MPI_Type_get_extent
MPI_Type_get_extent_x
MPI_Type_get_name
MPI_Type_get_true_extent
MPI_Type_get_true_extent_x
MPI_Type_hindexed
MPI_Type_hvector
MPI_Type_indexed
MPI_Type_lb
MPI_Type_match_size
MPI_Type_set_attr

OpenMP

High Performance Computing

The Serial Version of cpi

```
#include <stdio.h>
#include "gettime.h"

static long steps = 1000000000;

int main (int argc, const char *argv[]) {
    int i;
    double x;
    double pi;

    double step = 1.0/(double) steps;
    double sum = 0.0;
    double start = getTime();


    for (i=0; i < steps; i++) {
        x = (i+0.5)*step;
        sum += 4.0 / (1.0+x*x);
    }
    pi = step * sum;
    double delta = getTime() - start;
    printf("PI = %.16g computed in %.4g seconds\n", pi, delta);
}
```

Start of OpenMP Version

```
#include <stdio.h>
#include "gettime.h"

static long steps = 1000000000;

int main (int argc, const char *argv[]) {
    int i;
    double x;
    double pi;

    double step = 1.0/(double) steps;
    double sum = 0.0;
    double start = getTime();

#pragma omp parallel for
    for (i=0; i < steps; i++) {
        x = (i+0.5)*step;
        sum += 4.0 / (1.0+x*x);
    }
    pi = step * sum;
    double delta = getTime() - start;
    printf("PI = %.16g computed in %.4g seconds\n", pi, delta);
}
```

Need to Update our Makefile

```
CFLAGS=-Wall -O3 -ffast-math -mavx2 -fopenmp

CC=gcc

cpi             : cpi.o gettime.o

cpi.o           : cpi.c gettime.h

gettime.o       : gettime.c gettime.h

clean:
        rm -f cpi cpi.o gettime.o
```

Not quite good enough

```
dhcp-94-191:cpi3$ make
gcc -Wall -O3 -ffast-math -mavx2 -fopenmp    -c -o cpi.o cpi.c
gcc -Wall -O3 -ffast-math -mavx2 -fopenmp    -c -o gettime.o gettime.c
gcc   cpi.o gettime.o   -o cpi
Undefined symbols for architecture x86_64:
  "_GOMP_parallel", referenced from:
      _main in cpi.o
  "_omp_get_num_threads", referenced from:
      _main._omp_fn.0 in cpi.o
  "_omp_get_thread_num", referenced from:
      _main._omp_fn.0 in cpi.o
ld: symbol(s) not found for architecture x86_64
collect2: error: ld returned 1 exit status
make: *** [cpi] Error 1
```

Makefile need linker flags too

```
CFLAGS=-Wall -O3 -ffast-math -mavx2 -fopenmp
LDFLAGS=-fopenmp
CC=gcc

cpi             : cpi.o gettime.o

cpi.o           : cpi.c gettime.h

gettime.o       : gettime.c gettime.h

clean:
        rm -f cpi cpi.o gettime.o
```

Better. Not good, but better.

```
dhcp-94-191:cpi3$ make
gcc -fopenmp  cpi.o gettime.o   -o cpi
dhcp-94-191:cpi3$ ./cpi
PI = 0.5675882184166633 computed in 0.2785 seconds
```

(Laptop)

This is not pi

This is pi

```
dhcp-94-191:cpi3$ OMP_NUM_THREADS=1 ./cpi
PI = 3.141592653589729 computed in 0.5467 seconds
dhcp-94-191:cpi3$ OMP_NUM_THREADS=2 ./cpi
PI = 1.287002217586625 computed in 0.2806 seconds
dhcp-94-191:cpi3$ OMP_NUM_THREADS=4 ./cpi
PI = 0.9799146525073925 computed in 0.2782 seconds
dhcp-94-191:cpi3$ OMP_NUM_THREADS=4 ./cpi
PI = 0.5675882184166633 computed in 0.279 seconds
dhcp-94-191:cpi3$ OMP_NUM_THREADS=4 ./cpi
PI = 0.9799146525073925 computed in 0.2772 seconds
```

Where is the problem?

```
#include <stdio.h>
#include "gettime.h"

static long steps = 1000000000;

int main (int argc, const char *argv[]) {
    int i;
    double pi;


    double step = 1.0/(double) steps;
    double sum = 0.0;
    double start = getTime();

#pragma omp parallel for
    for (i=0; i < steps; i++) {
        double x = (i+0.5)*step;
        sum += 4.0 / (1.0+x*x);
    }
    pi = step * sum;
    double delta = getTime() - start;
    printf("PI = %.16g computed in %.4g seconds\n", pi, delta);
}
```

Where is the problem?

```
#include <stdio.h>
#include "gettime.h"

static long steps = 1000000000;

int main (int argc, const char *argv[]) {
    int i;
    double pi;


    double step = 1.0/(double) steps;
    double sum = 0.0;
    double start = getTime();

#pragma omp parallel for reduction(+ : sum)
    for (i=0; i < steps; i++) {
        double x = (i+0.5)*step;
        sum += 4.0 / (1.0+x*x);
    }
    pi = step * sum;
    double delta = getTime() - start;
    printf("PI = %.16g computed in %.4g seconds\n", pi, delta);
}
```

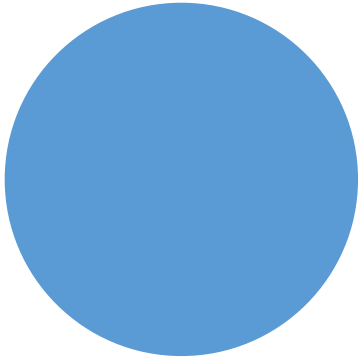
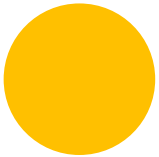
Like MPI_Reduce

All others must be “handled”

Loop variable is fine

Now it all works

```
dhcp-94-191:cpi3$ make
gcc -Wall -O3 -ffast-math -mavx2 -fopenmp    -c -o cpi.o cpi.c
gcc -fopenmp  cpi.o gettime.o   -o cpi

dhcp-94-191:cpi3$ ./cpi
PI = 3.141592653589774 computed in 0.2805 seconds

dhcp-94-191:cpi3$ OMP_NUM_THREADS=4 ./cpi
PI = 3.141592653589774 computed in 0.2776 seconds

dhcp-94-191:cpi3$ OMP_NUM_THREADS=3 ./cpi
PI = 3.141592653589908 computed in 0.2786 seconds

dhcp-94-191:cpi3$ OMP_NUM_THREADS=2 ./cpi
PI = 3.141592653589974 computed in 0.2841 seconds

dhcp-94-191:cpi3$ OMP_NUM_THREADS=20 ./cpi
PI = 3.141592653589802 computed in 0.2766 seconds
```

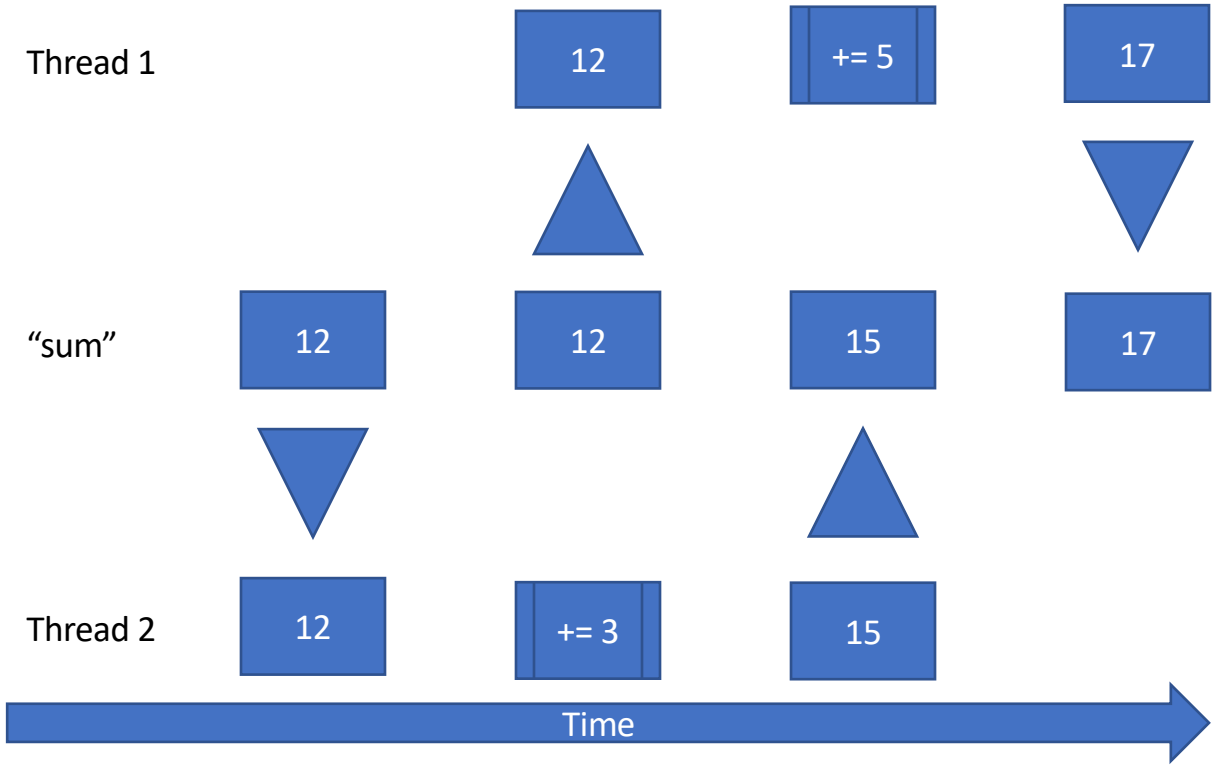



Parallel Pitfalls

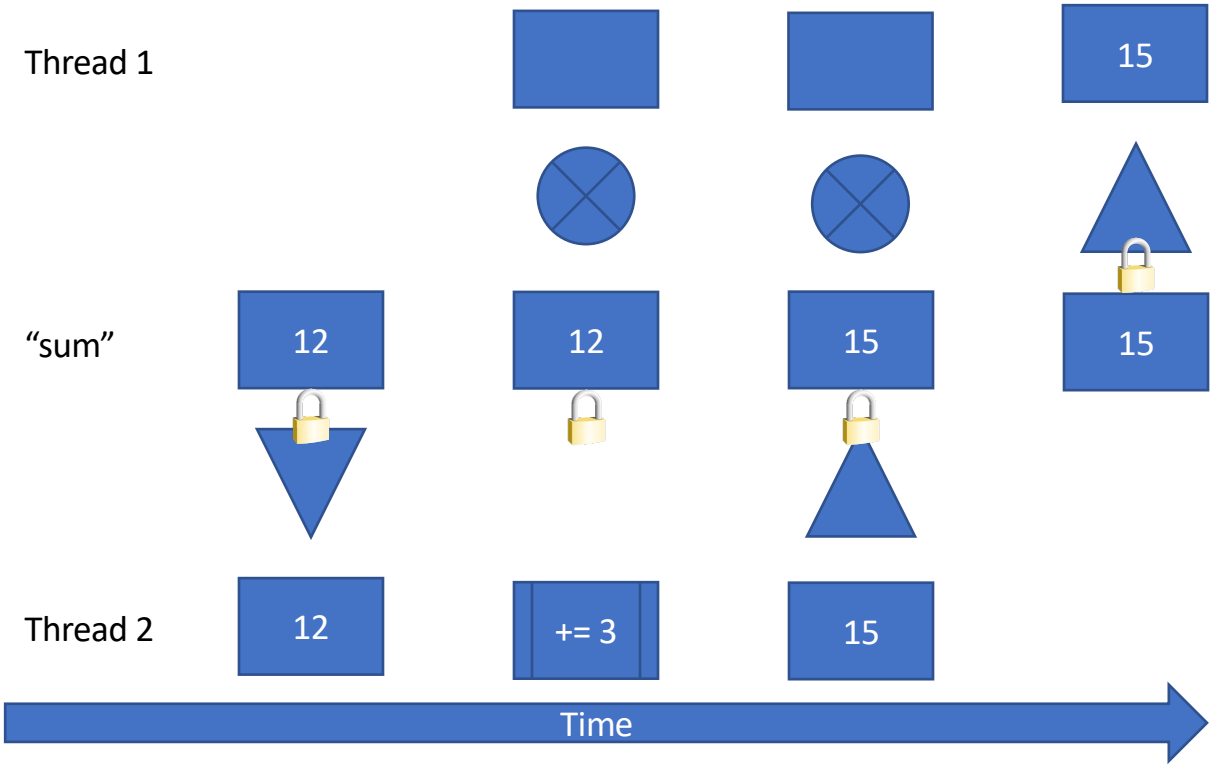
What you're glad that you don't need to know

(too much)

Critical Section: sum += value



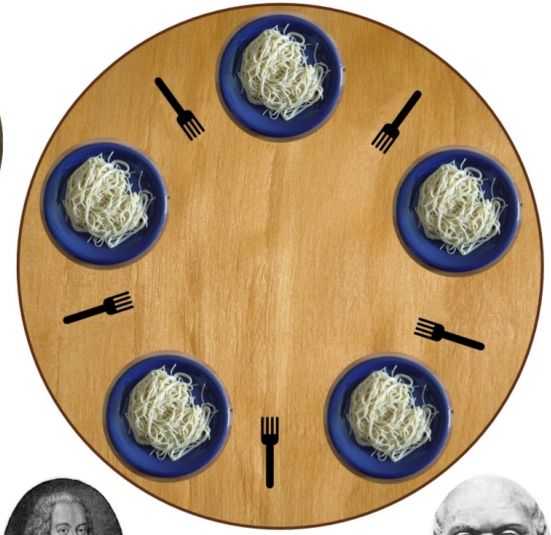
Solution: Locking

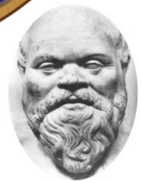


Deadlock

- Think for some time
- Pick up left fork
- Pick up right fork
- Eat for some time
- Put down right fork
- Put down left fork
- Continue Thinking

Edsger W. Dijkstra





Requirements for a Deadlock

Mutual Exclusion

- Resource cannot be shared

Hold and Wait

- Must hold a resource while waiting for another

No Pre-emption

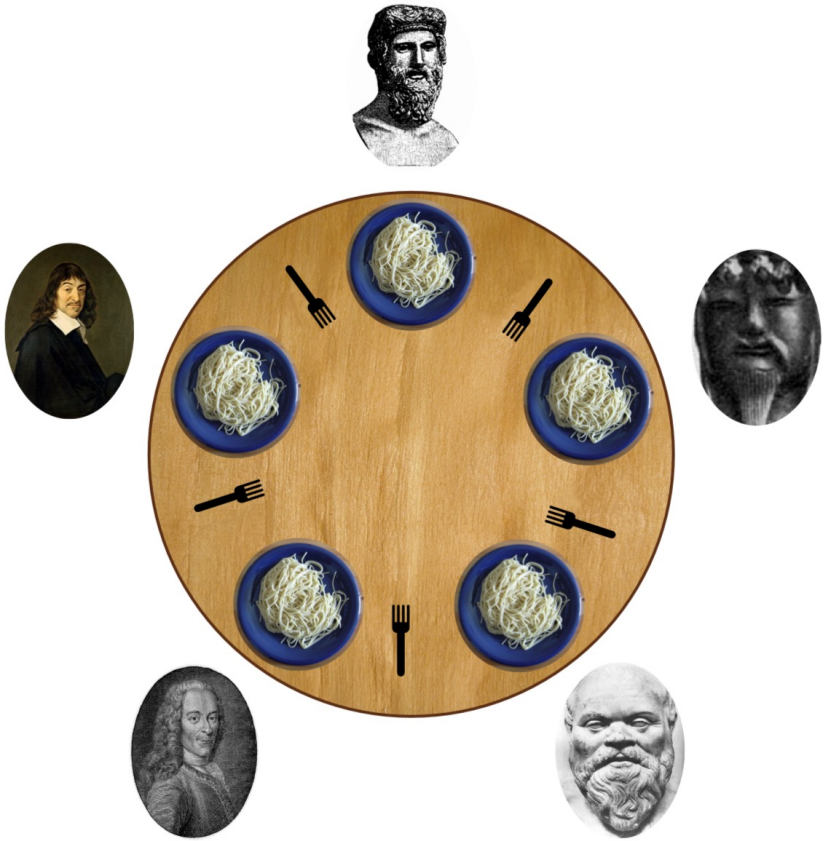
- Held resources cannot be given away

Circular Wait

- The wait dependency must be circular

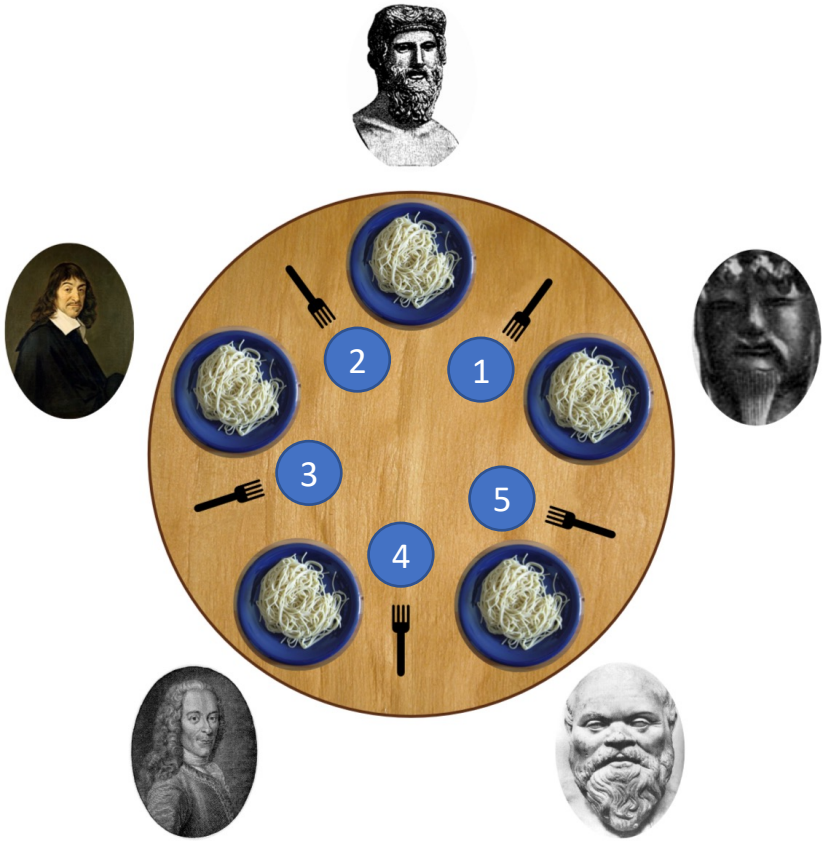
Starvation
(No Hold and Wait)

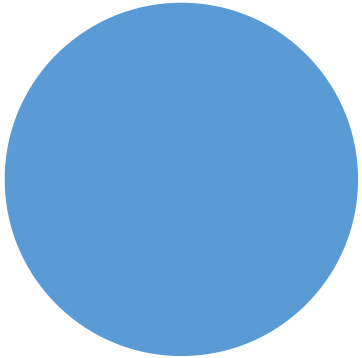
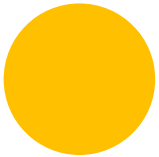
- Think for some time
- Pick up left fork
- Right fork not available
- … or left fork not available
- Put down left fork
- Think for a while
- Eventually try again



Resource Hierarchy
(No Circular Wait)

- Number each resource
- Choose resources in order
  - Confucius is different
- Cannot deadlock
- Cannot starve?

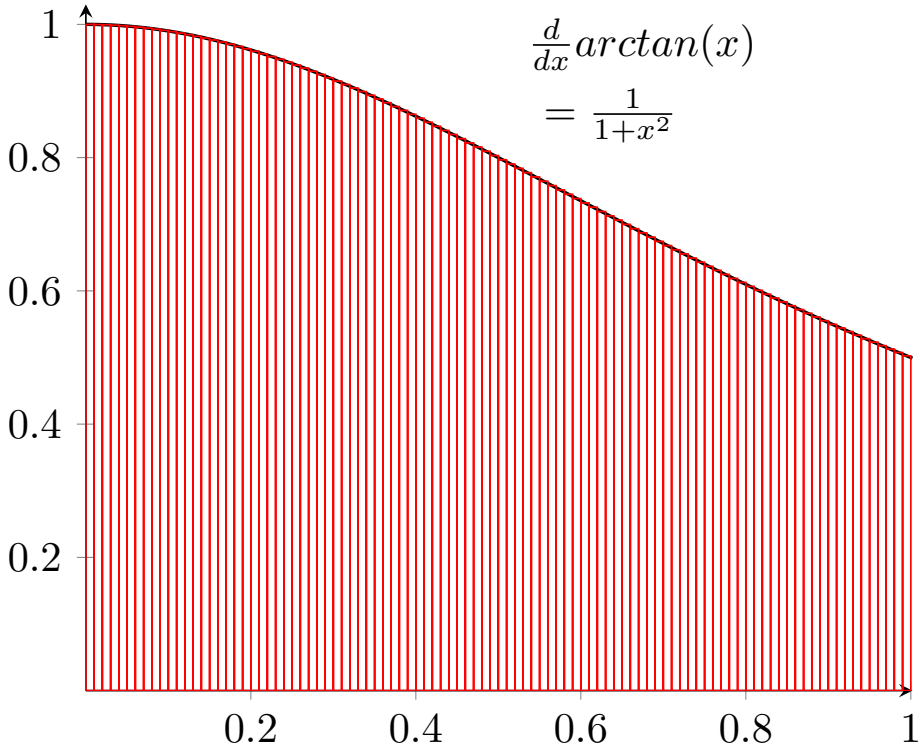


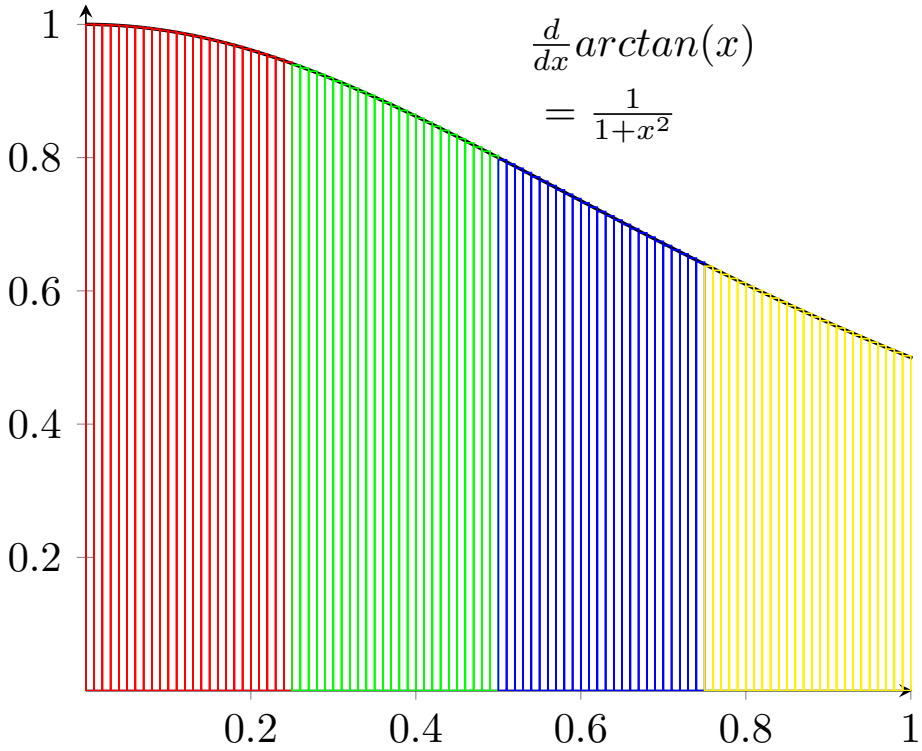
Load Balancing

How to divide up the work

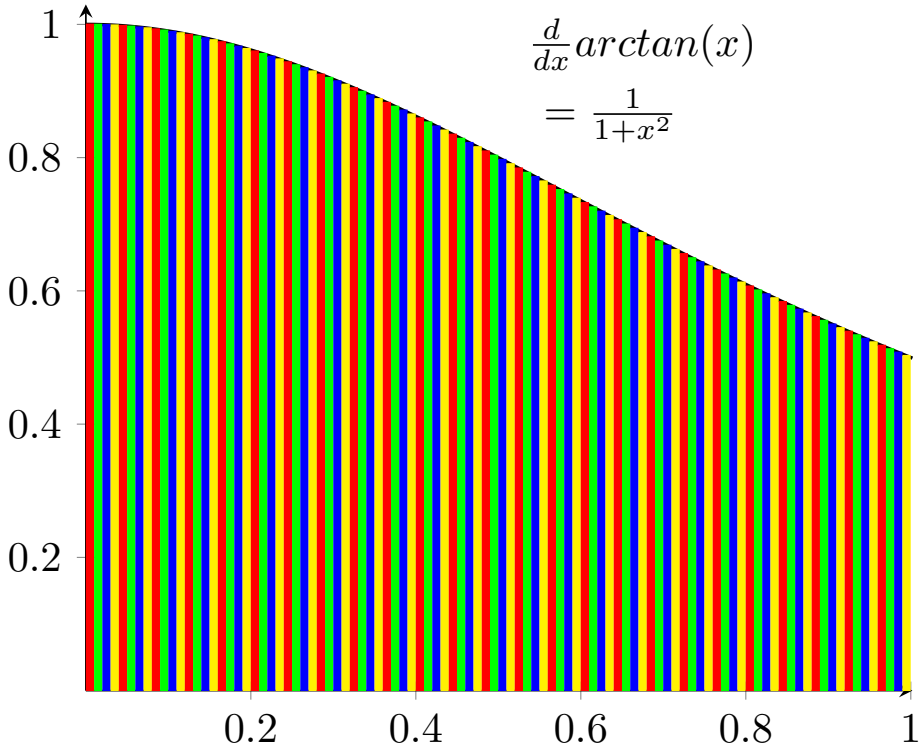
Divide 100 rectangles into four domains



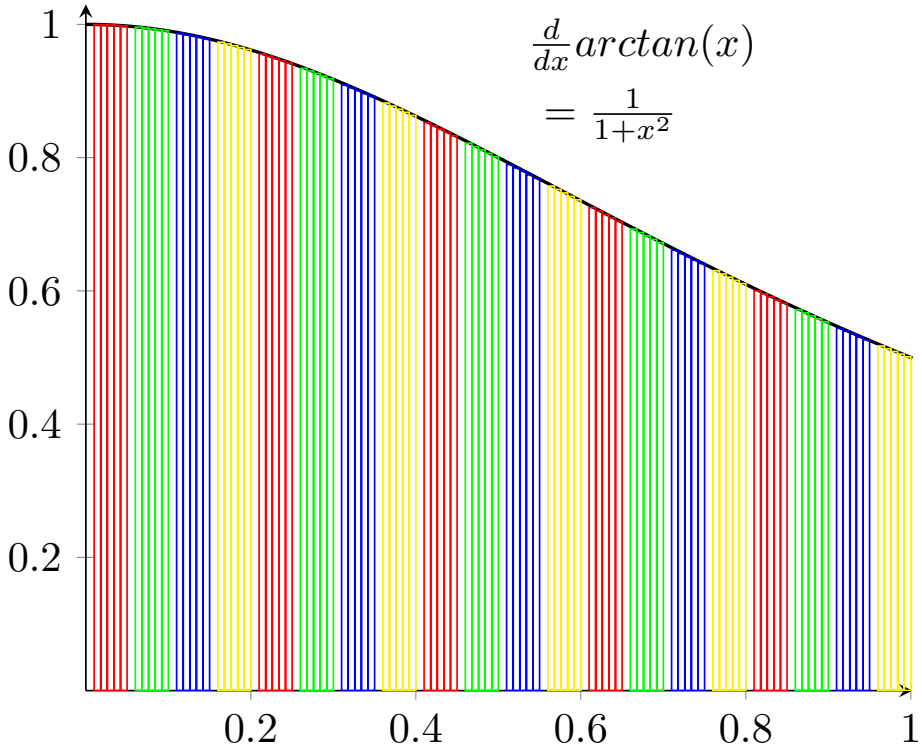
Option 1



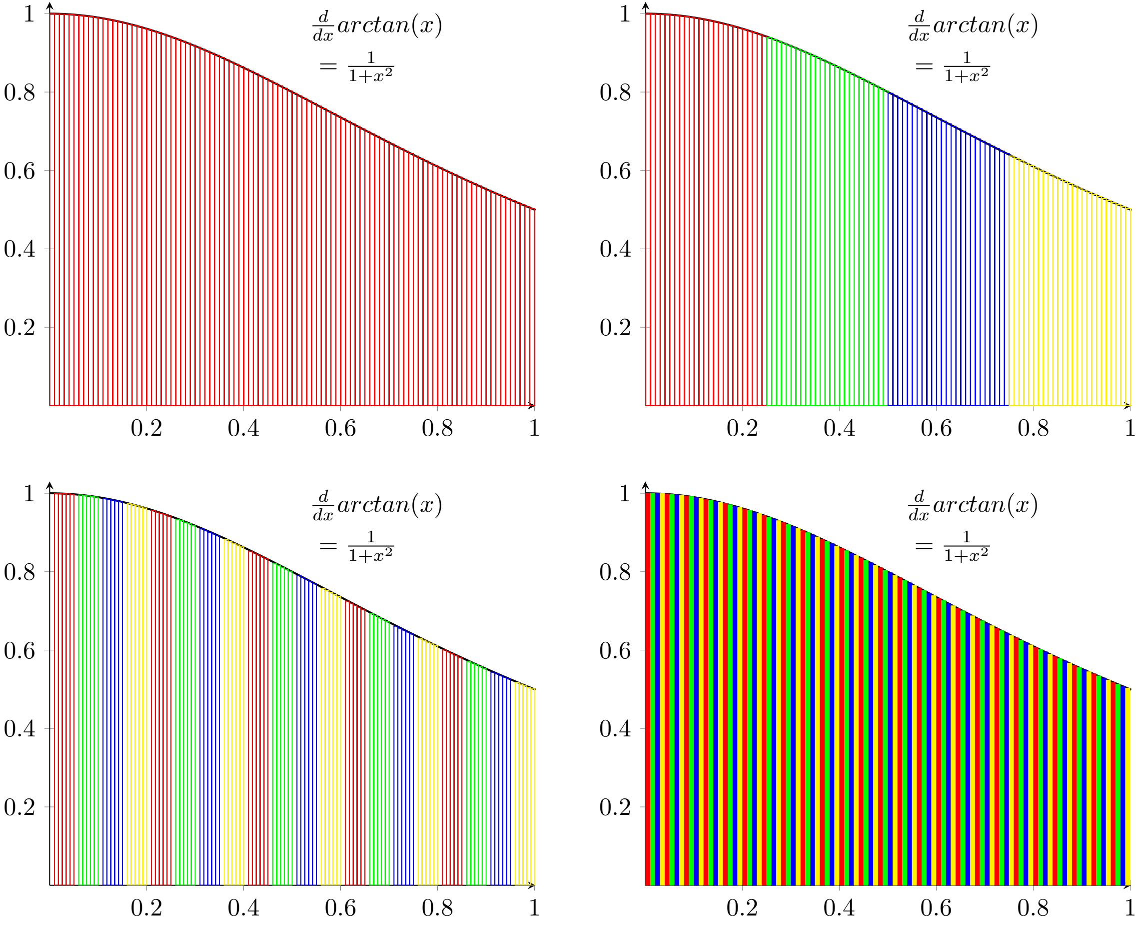
Option 2



Option 3



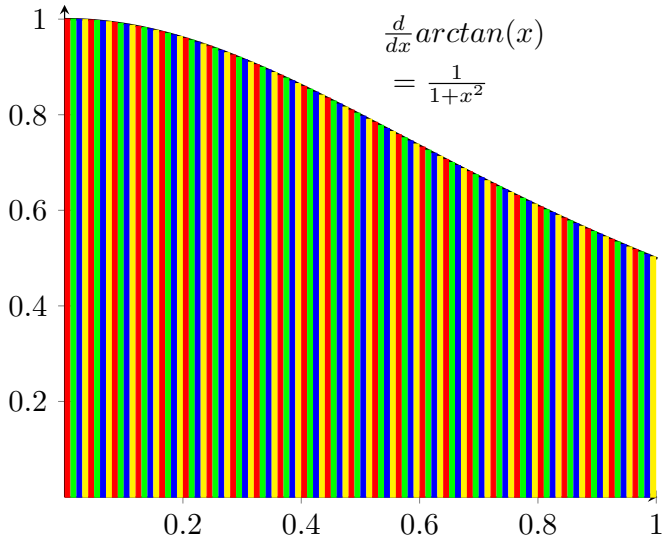
Which is the best choice?

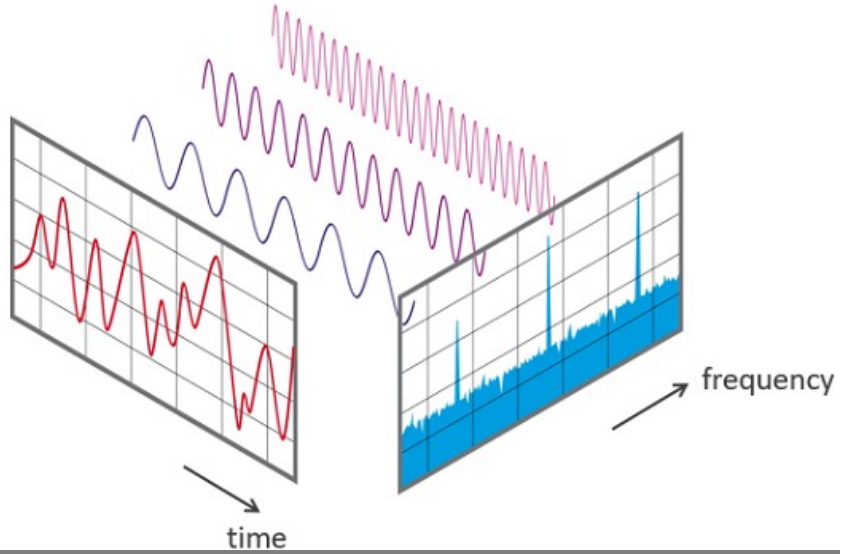


University of Zurich[UZH]

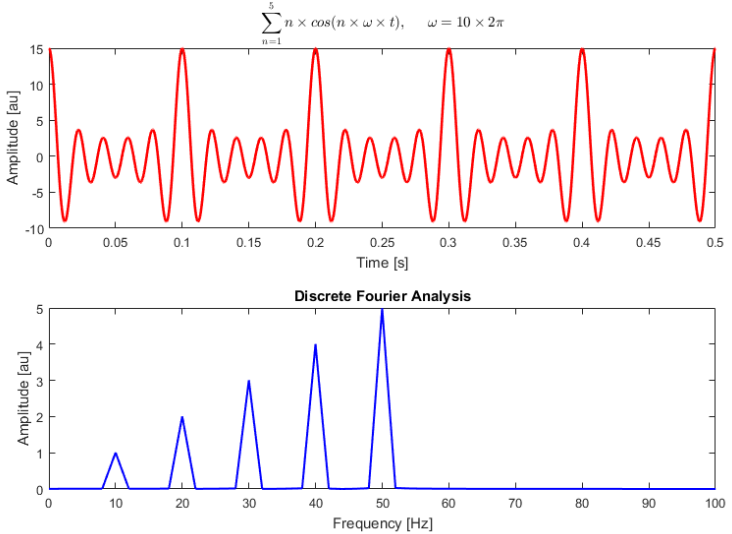
The MPI version of cpi

```
MPI_Init(&argc,&argv);                          /* Connect processes to each other */
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);    /* Get total number of processes */
MPI_Comm_rank(MPI_COMM_WORLD,&myid);                    /* Rank of this process */
MPI_Get_processor_name(name, &resultlen);

printf("This is Process-%d/%d running on %s \n",myid,numprocs,name);
MPI_Barrier(MPI_COMM_WORLD);

if(myid == 0) {
    printf("This program uses %d processes\n", numprocs);
    n = 1000000000;
    }

MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);/* Send n from 0 to all others */

sum = 0.0;
h   = 1.0/n;
for (i=myid+0.5; i<n; i+=numprocs) {
    sum += dx_arctan(i*h);
    }
mypi = 4.0*h*sum;

                    /* Add all partial sums to each other and send to rank 0 */
MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

if (myid == 0) {
    printf("pi is approximately %.16f\n",pi);
    }
MPI_Finalize();                                 /* Disconnect all processes */
```



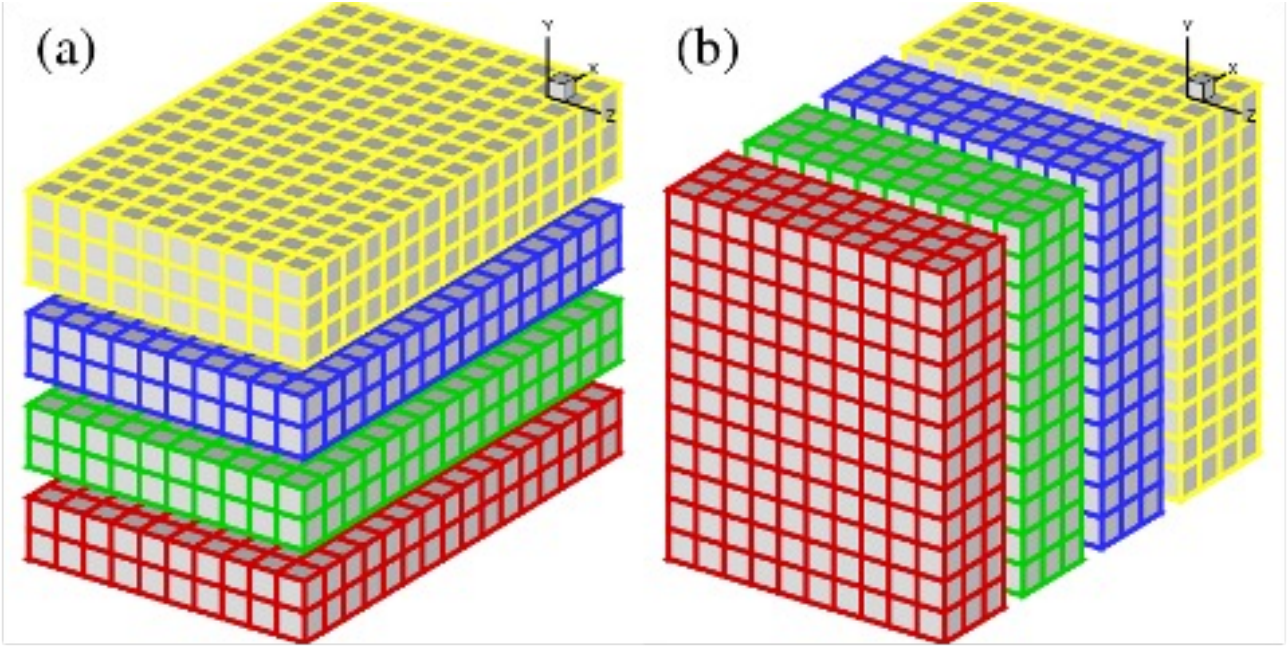


By Phonical

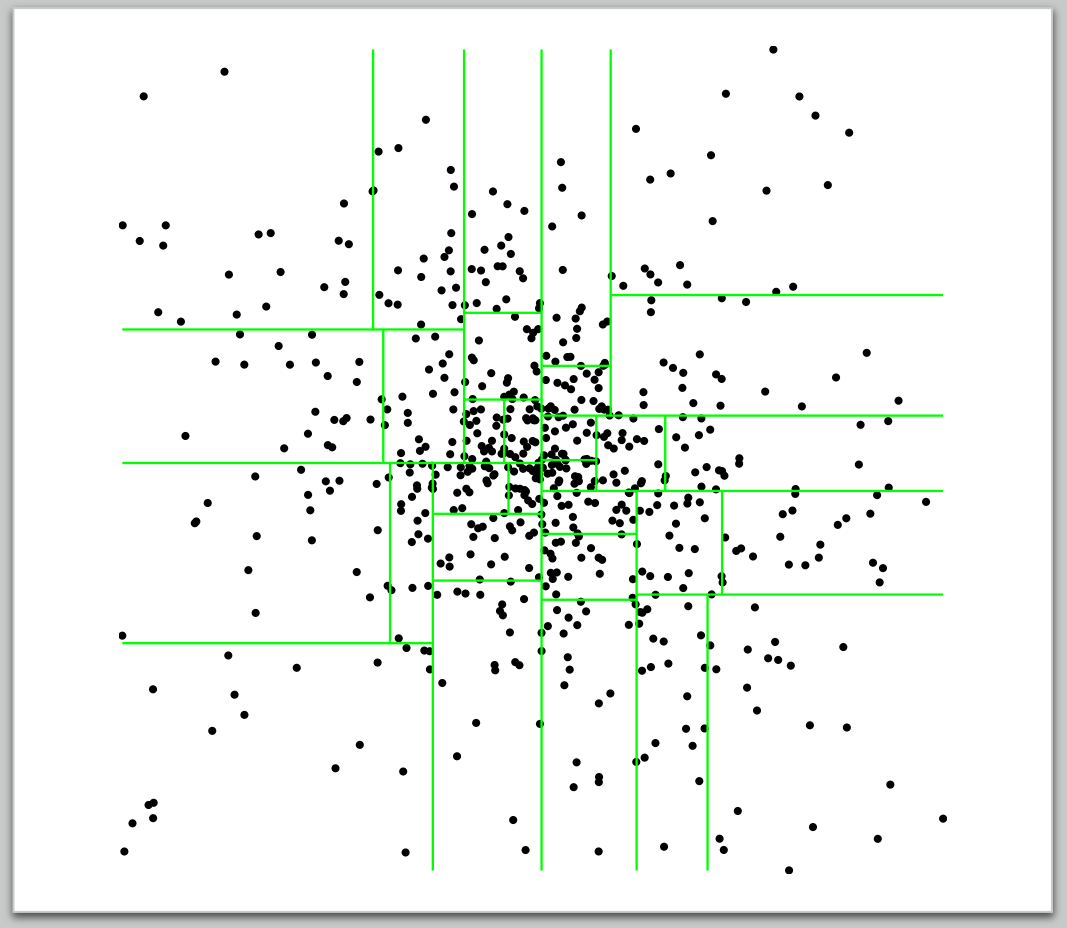


By DaveSGage

Fast Fourier Transform



http://2decomp.org



Particle Data (N-Body)