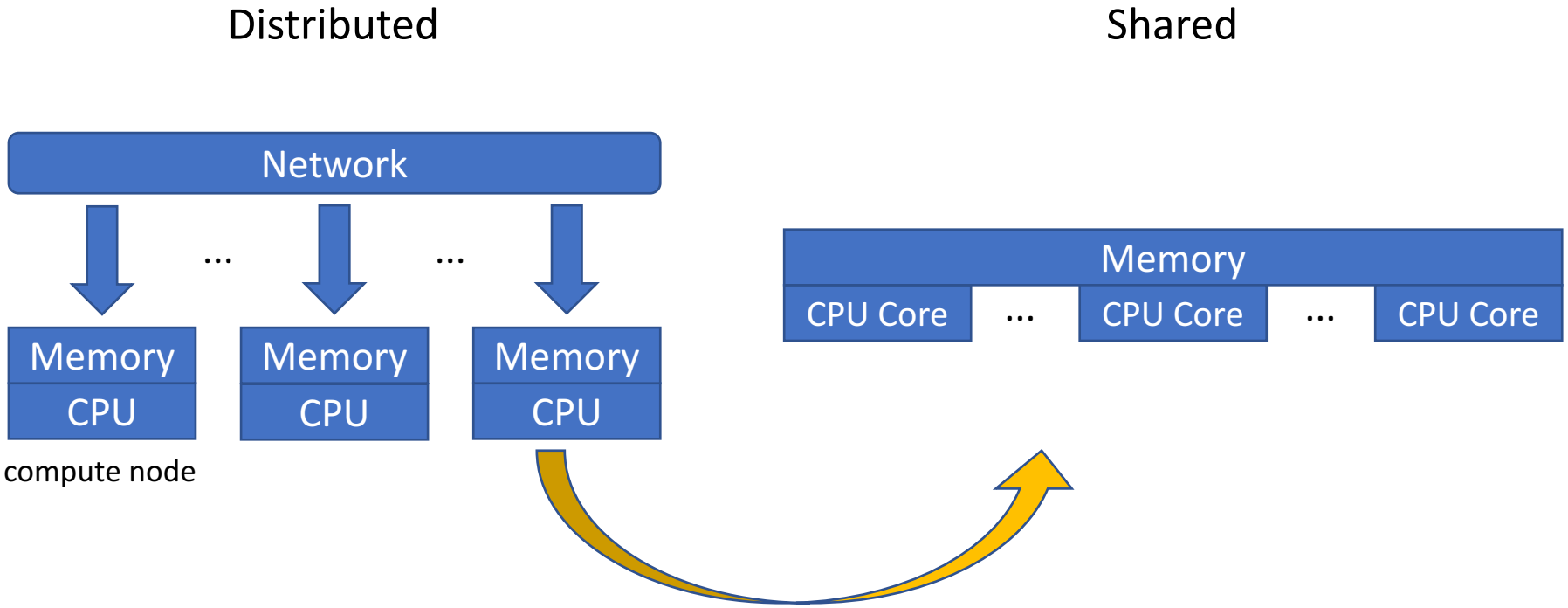
OpenMP 101

Junwen Li

RECSEM REU Site Program
May 31, 2018

Joint Institute for Computational Sciences
THE UNIVERSITY OF TENNESSEE KNOXVILLE
THE UNIVERSITY OF TENNESSEE HEALTH SCIENCE CENTER

Memory types



MPI

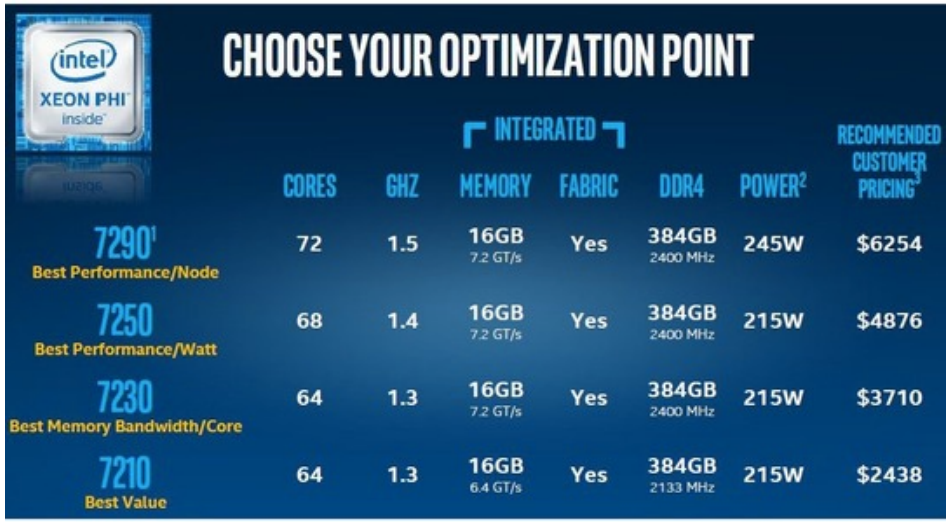
Develop parallel programs from the very beginning

OpenMP

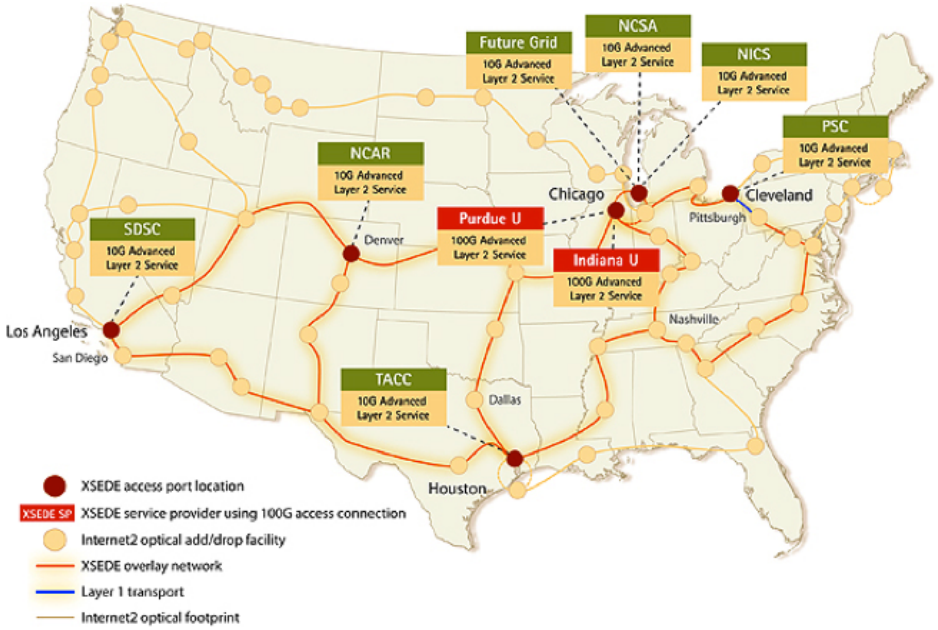
Incrementally parallelize serial programs

Computing resources good for OpenMP

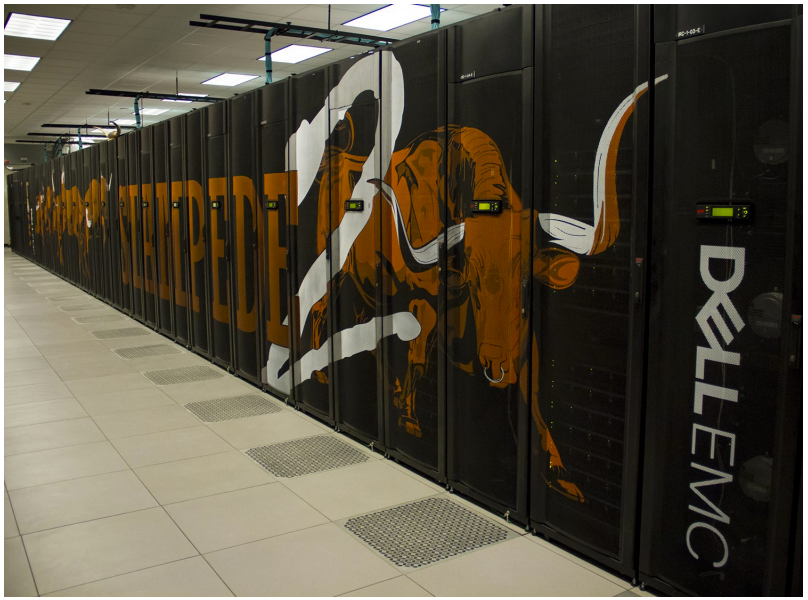
Node Set	CPU	Nodes	Cores/node	GB Mem/Node	Total Cores	Interconnect
Beacon	Intel® Xeon® E5-2670	43	16	256	704	FDR
Rho	Intel® Xeon® E5-2670	48	16	32	768	QDR
Sigma	Intel® Xeon® E5-2680	118	24	128	2832	FDR
Monster	Intel® Xeon® E5-2687W	1	24	1024	24	Ethernet
KNL	Intel® Xeon® Phi® 7210	4	64	256	256	EDR



Computing resources good for OpenMP



https://sciencenode.org/feature/autobahn-xsede-users.php



https://portal.tacc.utexas.edu/user-guides/stampede2

Stampede2 hosts 4,200 KNL compute nodes

Model:	Intel Xeon Phi 7250 ("Knights Landing")
Total cores per KNL node:	68 cores on a single socket

We will talk about …

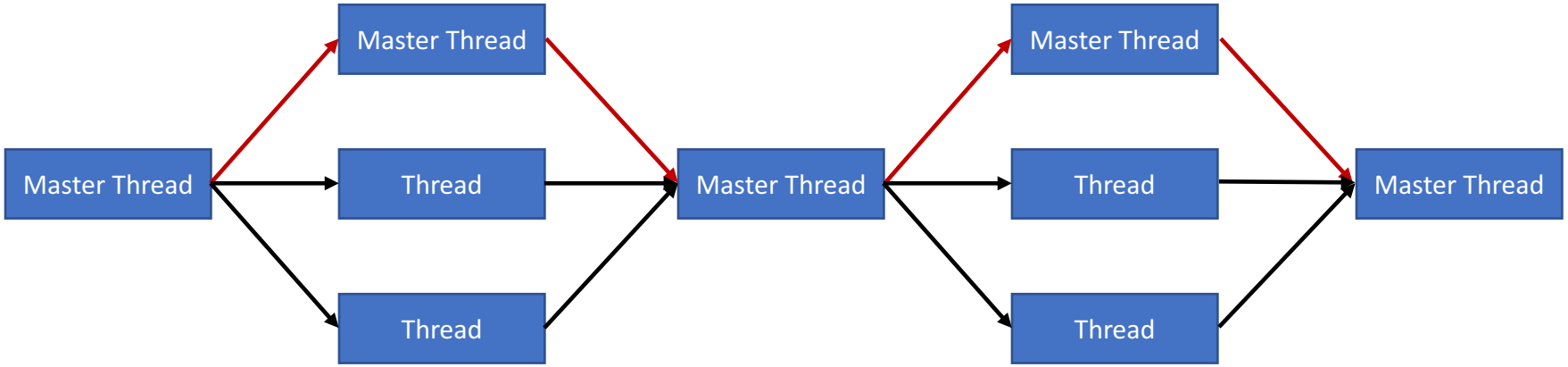
- Fork/join model
- OpenMP syntax and directives
  - Hello world example
  - π calculation
  - Many loops or nested loops
- Real research example
  - Optical properties of solar cell materials
  - Five lines to parallelize the code

BIG ORANGE BIG IDEAS

Fork / join model of OpenMP framework

- Serial program

- OpenMP program



Serial “Hello World” in C/C++

```
#include <stdio.h>

int main()
{
   int  nthread = 1;
   int  ithread = 1;
   printf(“Hello world from %d of %d!\n”, ithread, nthread);

   return 0;
}
```

```
$ ./hello_serial_c.x
Hello World from 1 of 1!
```


BIG ORANGE BIG IDEAS

OpenMP “Hello World” in C/C++

Runtime Library

Compiler Directives

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

int main()
{
  omp_set_num_threads(4);
#pragma omp parallel
{
   int  nthread = omp_get_num_threads();
   int  ithread = omp_get_thread_num();
   printf(“Hello world from %d of %d.\n”, ithread, nthread);
 }
   return 0;
}
```

```
$ ./hello_omp_c.x
Hello World from 0 of 4!
Hello World from 3 of 4!
Hello World from 1 of 4!
Hello World from 2 of 4!
```

Thread index starts with 0!

BIG ORANGE BIG IDEAS

How to compile OpenMP code

Compiler Type	Compiler Command	Compiler Flag
Intel	icc	-qopenmp
	icpc	-qopenmp
	ifort	-qopenmp
GNU	gcc	-fopenmp
	g++	-fopenmp
	gfortran	-fopenmp

```
$icc -qopenmp -o hello.exe hello.c
$./hello.exe
```


BIG ORANGE BIG IDEAS

Two ways to specify the number of threads

Inside the source code

omp_set_num_threads(4);

```
$ ./hello_omp_c.x
Hello World from 0 of 4!
Hello World from 3 of 4!
Hello World from 1 of 4!
Hello World from 2 of 4!
```

omp_set_num_threads(6);

```
./hello_omp_c.x
Hello World from 1 of 6!
Hello World from 0 of 6!
Hello World from 3 of 6!
Hello World from 2 of 6!
Hello World from 5 of 6!
Hello World from 4 of 6!
```

Outside the source code

export OMP_NUM_THREADS = 4

export OMP_NUM_THREADS = 6

```
for ((n=1; n<=16; n++))
do
export OMP_NUM_THREADS=$n
/usr/bin/time -f "%e" OPENMP_CODE.EXE
done
```


BIG ORANGE BIG IDEAS

Three core components in OpenMP

Compiler Directives	Runtime Library	Environment Variables
#pragma omp parallel	omp_set_number_threads()	OMP_NUM_THREADS
…	…	…
Directives • added to a serial program • interpreted at compile time	Directives • added to a serial program • executed at run time	Directives • added after compile time • used at run time

BIG ORANGE BIG IDEAS

Example: π calculation

$$\int_0^1 \frac{4}{1+x^2}\,dx = \pi$$

$$\sum_{i=1}^{N} \frac{4}{1+x_i^2}\Delta \cong \pi$$

$$\Delta = 1/N$$

$$x_i = (i - 0.5)\Delta$$

```
#define NSTEPS 2000

int main()
{
  double pi, sum, step, x;
  long int i;

  sum = 0;
  step = 1.0 / (double)NSTEPS;
  for (i=1; i<=NSTEPS; i++)
    {
      x = step * ((double)i-0.5);
      sum += 4.0/(1.0+x*x);
    }
  pi = sum * step;

  printf("Pi is %4.12f\n", pi);

  return 0;
}
```

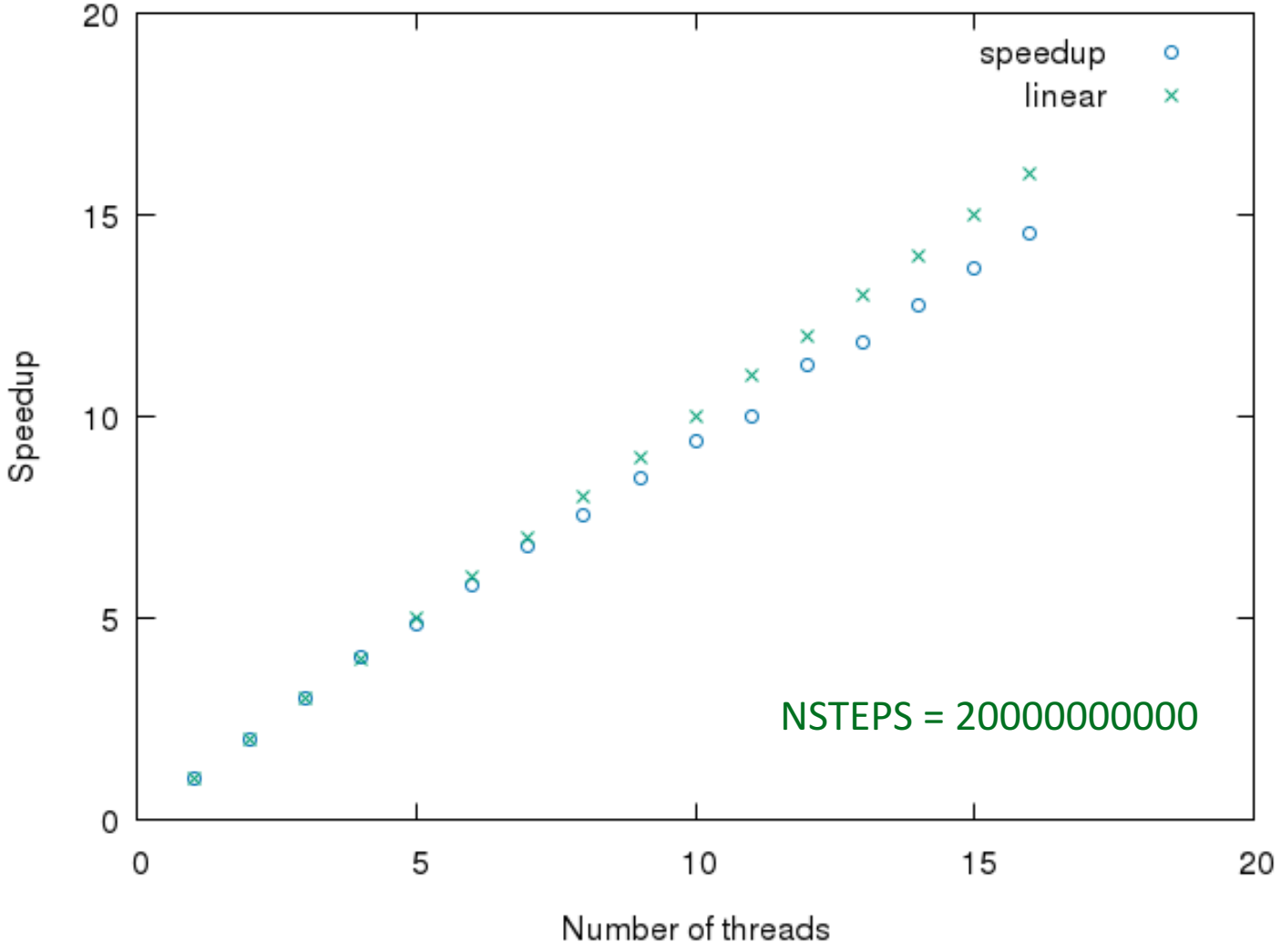

BIG ORANGE BIG IDEAS

OpenMP version of π calculation

```
#define NSTEPS 2000

int main()
{
  double pi, sum, step, x;
  long int i;

  sum = 0;
  step = 1.0 / (double)NSTEPS;

#pragma omp parallel
#pragma omp for private(i,x) reduction(+:sum)
  for (i=1; i<=NSTEPS; i++)
    {
      x = step * ((double)i-0.5);
      sum += 4.0/(1.0+x*x);
    }
  pi = sum * step;

  printf("Pi is %4.12f\n", pi);

  return 0;
}
```

```
#define NSTEPS 2000

int main()
{
  double pi, sum, step, x;
  long int i;

  sum = 0;
  step = 1.0 / (double)NSTEPS;

#pragma omp parallel for private(i,x) reduction(+:sum)
  for (i=1; i<=NSTEPS; i++)
    {
      x = step * ((double)i-0.5);
      sum += 4.0/(1.0+x*x);
    }
  pi = sum * step;

  printf("Pi is %4.12f\n", pi);

  return 0;
}
```

Speedup vs number of threads



for construct -- private

```
#define NSTEPS 2000

int main()
{
  double pi, sum, step, x;
  long int i;

  sum = 0;
  step = 1.0 / (double)NSTEPS;

#pragma omp parallel for private(i,x) reduction(+:sum)
  for (i=1; i<=NSTEPS; i++)
    {
      x = step * ((double)i-0.5);
      sum += 4.0/(1.0+x*x);
    }
  pi = sum * step;

  printf("Pi is %4.12f\n", pi);

  return 0;
}
```

OMP_NUM_THREADS = 2

Thread 0

```
double x0; long int i0;

for (i0=1; i0<=1000; i0++)
{
   x0 = step * ((double)i0 - 0.5);
   ...
}
```

Thread 1

```
double x1; long int i1;

for (i1=1001; i1<=2000; i1++)
{
 x1 = step * ((double)i1 - 0.5);
 ...
}
```


BIG ORANGE BIG IDEAS

for construct -- reduction

```
#define NSTEPS 2000

int main()
{
  double pi, sum, step, x;
  long int i;

  sum = 0;
  step = 1.0 / (double)NSTEPS;

#pragma omp parallel for private(i,x) reduction(+:sum)
  for (i=1; i<=NSTEPS; i++)
    {
      x = step * ((double)i-0.5);
      sum += 4.0/(1.0+x*x);
    }
  pi = sum * step;

  printf("Pi is %4.12f\n", pi);

  return 0;
}
```

OMP_NUM_THREADS = 2

Thread 0

```
double x0; long int i0;
double sum0;
for (i0=1; i0<=1000; i0++)
{
  x0 = step * ((double)i0 - 0.5);
  sum0 += 4.0/(1.0+x0*x0);
}
```

Thread 1

```
double x1; long int i1;
double sum1;
for (i1=1001; i1<=2000; i1++)
{
  x1 = step * ((double)i1 - 0.5);
  sum1 += 4.0/(1.0+x1*x1);
}
```

```
sum += sum0 + sum1;
```


BIG ORANGE BIG IDEAS

for construct -- reduction

```
#define NSTEPS 2000

int main()
{
  double pi, sum, step, x;
  long int i;

  sum = 0;
  step = 1.0 / (double)NSTEPS;

#pragma omp parallel for private(i,x) reduction(+:sum)
  for (i=1; i<=NSTEPS; i++)
    {
      x = step * ((double)i-0.5);
      sum += 4.0/(1.0+x*x);
    }
  pi = sum * step;

  printf("Pi is %4.12f\n", pi);

  return 0;
}
```

OMP_NUM_THREADS = 2

Thread 0

```
double x0; long int i0;
double sum0 = 0;
for (i0=1; i0<=1000; i0++)
{
   x0 = step * ((double)i0 - 0.5);
   sum0 += 4.0/(1.0+x0*x0);
}
```

Thread 1

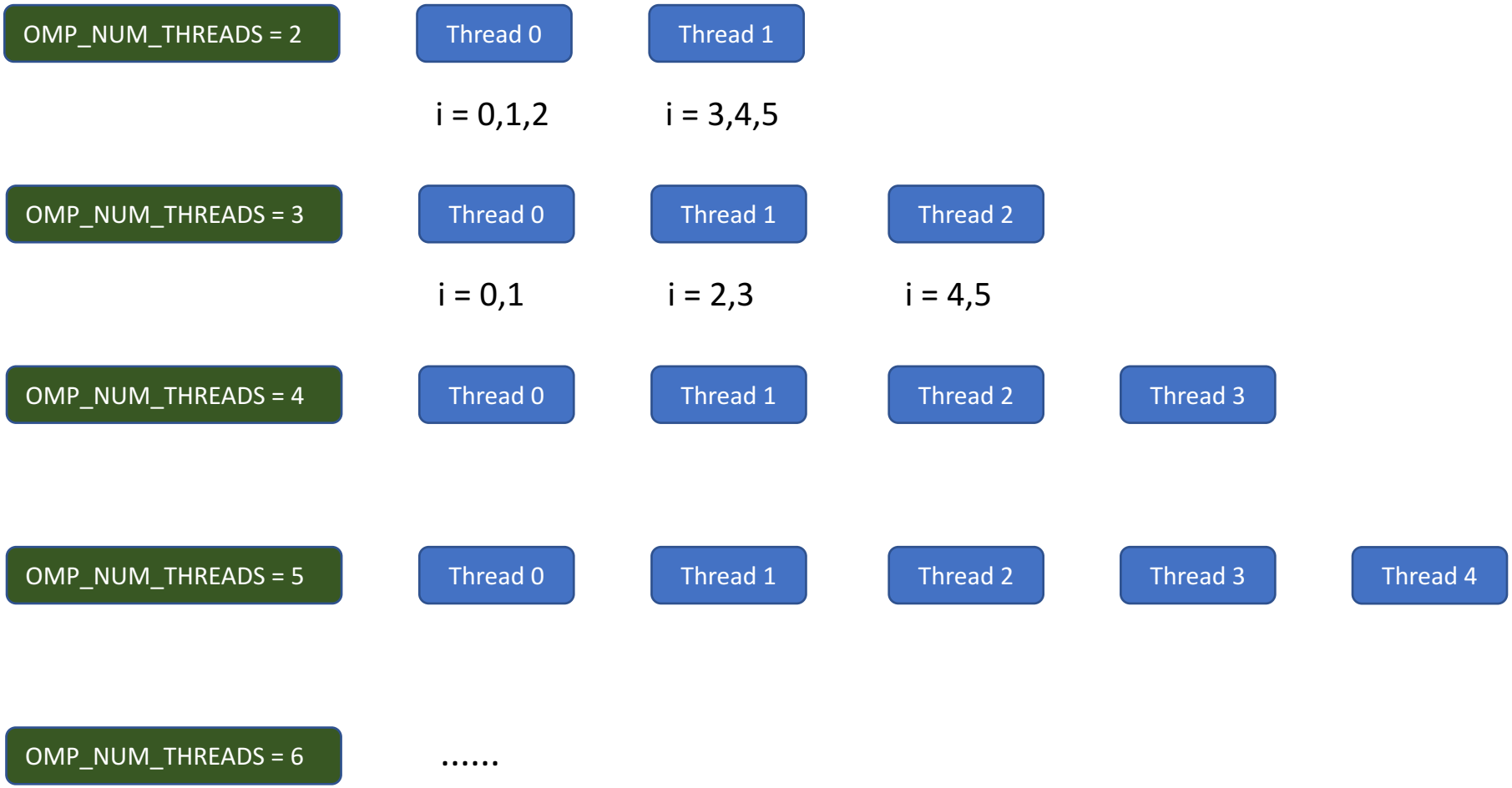
```
double x1; long int i1;
double sum1 = 0;
for (i1=1001; i1<=2000; i1++)
{
   x1 = step * ((double)i1 - 0.5);
   sum1 += 4.0/(1.0+x1*x1);
}
```

```
sum += sum0 + sum1;
```


BIG ORANGE BIG IDEAS

Static loop iteration assignment

```
#pragma omp parallel for
    for (i = 0; i < 6; i++)
```



Static loop iteration assignment

```
#pragma omp parallel for
    for (i = 0; i < 6; i++)
```

OMP_NUM_THREADS = 2	Thread 0	Thread 1			
	i = 0,1,2	i = 3,4,5			
OMP_NUM_THREADS = 3	Thread 0	Thread 1	Thread 2		
	i = 0,1	i = 2,3	i = 4,5		
OMP_NUM_THREADS = 4	Thread 0	Thread 1	Thread 2	Thread 3	
	i = 0,1	i = 2,3	i = 4	i = 5	
OMP_NUM_THREADS = 5	Thread 0	Thread 1	Thread 2	Thread 3	Thread 4
	i = 0,1	i = 2	i = 3	i = 4	i = 5
OMP_NUM_THREADS = 6	.....				

BIG ORANGE BIG IDEAS

Static loop iteration assignment

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


int main ()
{
  int i, N;
  N = 6;


  #pragma omp parallel
  {
    #pragma omp for
    for (i = 0; i < 6; i++)
      printf("Loop 1: iter %3d, thread_id %3d\n", i, omp_get_thread_num());


    #pragma omp for
    for (i = 0; i < 6; i++)
      printf("Loop 2: iter %3d, thread_id %3d\n", i, omp_get_thread_num());
  }


  return 0;
}
```


BIG ORANGE BIG IDEAS

Static loop iteration assignment

```
export OMP_NUM_THREADS=6
```

First Run

```
Loop 1: iter   0, thread_id   0
Loop 1: iter   1, thread_id   1
Loop 1: iter   2, thread_id   2
Loop 1: iter   3, thread_id   3
Loop 1: iter   4, thread_id   4
Loop 1: iter   5, thread_id   5
Loop 2: iter   0, thread_id   0
Loop 2: iter   1, thread_id   1
Loop 2: iter   2, thread_id   2
Loop 2: iter   3, thread_id   3
Loop 2: iter   4, thread_id   4
Loop 2: iter   5, thread_id   5
```

Second Run

```
Loop 1: iter   0, thread_id   0
Loop 1: iter   1, thread_id   1
Loop 1: iter   2, thread_id   2
Loop 1: iter   3, thread_id   3
Loop 1: iter   4, thread_id   4
Loop 1: iter   5, thread_id   5
Loop 2: iter   0, thread_id   0
Loop 2: iter   1, thread_id   1
Loop 2: iter   2, thread_id   2
Loop 2: iter   3, thread_id   3
Loop 2: iter   4, thread_id   4
Loop 2: iter   5, thread_id   5
```


BIG ORANGE BIG IDEAS

Dynamic loop iteration assignment

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


int main ()
{
  int i, N;
  N = 6;


  #pragma omp parallel
  {
    #pragma omp for schedule(dynamic,1)
    for (i = 0; i < 6; i++)
      printf("Loop 1: iter %3d, thread_id %3d\n", i, omp_get_thread_num());


    #pragma omp for schedule(dynamic,1)
    for (i = 0; i < 6; i++)
      printf("Loop 2: iter %3d, thread_id %3d\n", i, omp_get_thread_num());
  }


  return 0;
}
```

export OMP_SCHEDULE=“dynamic,1”

BIG ORANGE BIG IDEAS

Dynamic loop iteration assignment

```
export OMP_NUM_THREADS=6
```

First Run

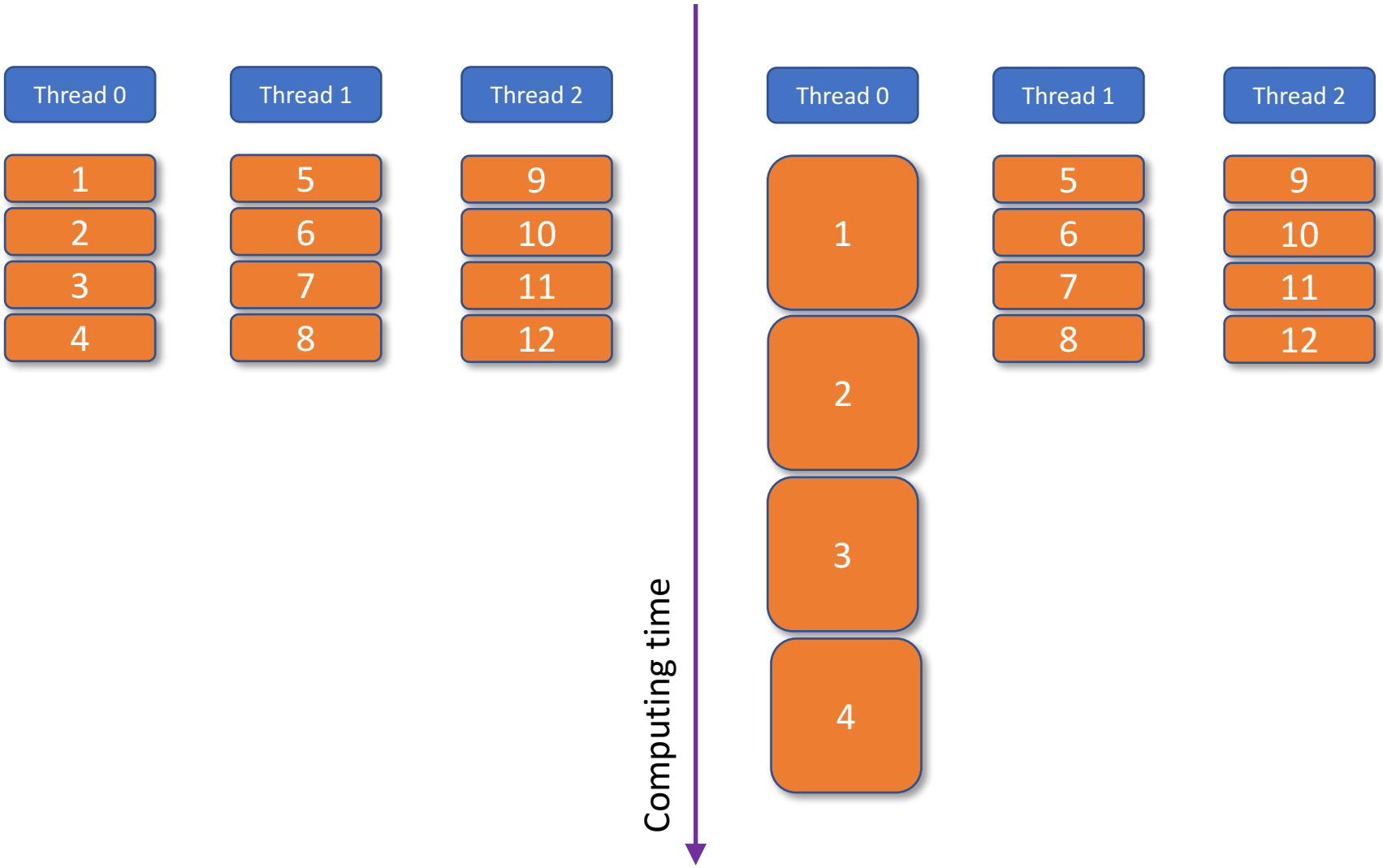
```
Loop 1: iter   0, thread_id   2
Loop 1: iter   1, thread_id   0
Loop 1: iter   2, thread_id   1
Loop 1: iter   3, thread_id   2
Loop 1: iter   4, thread_id   2
Loop 1: iter   5, thread_id   2
Loop 2: iter   0, thread_id   4
Loop 2: iter   1, thread_id   3
Loop 2: iter   2, thread_id   5
Loop 2: iter   3, thread_id   2
Loop 2: iter   4, thread_id   0
Loop 2: iter   5, thread_id   1
```

Second Run

```
Loop 1: iter   0, thread_id   0
Loop 1: iter   1, thread_id   3
Loop 1: iter   2, thread_id   2
Loop 1: iter   3, thread_id   1
Loop 1: iter   4, thread_id   4
Loop 1: iter   5, thread_id   1
Loop 2: iter   0, thread_id   5
Loop 2: iter   1, thread_id   2
Loop 2: iter   2, thread_id   1
Loop 2: iter   3, thread_id   3
Loop 2: iter   4, thread_id   4
Loop 2: iter   5, thread_id   0
```

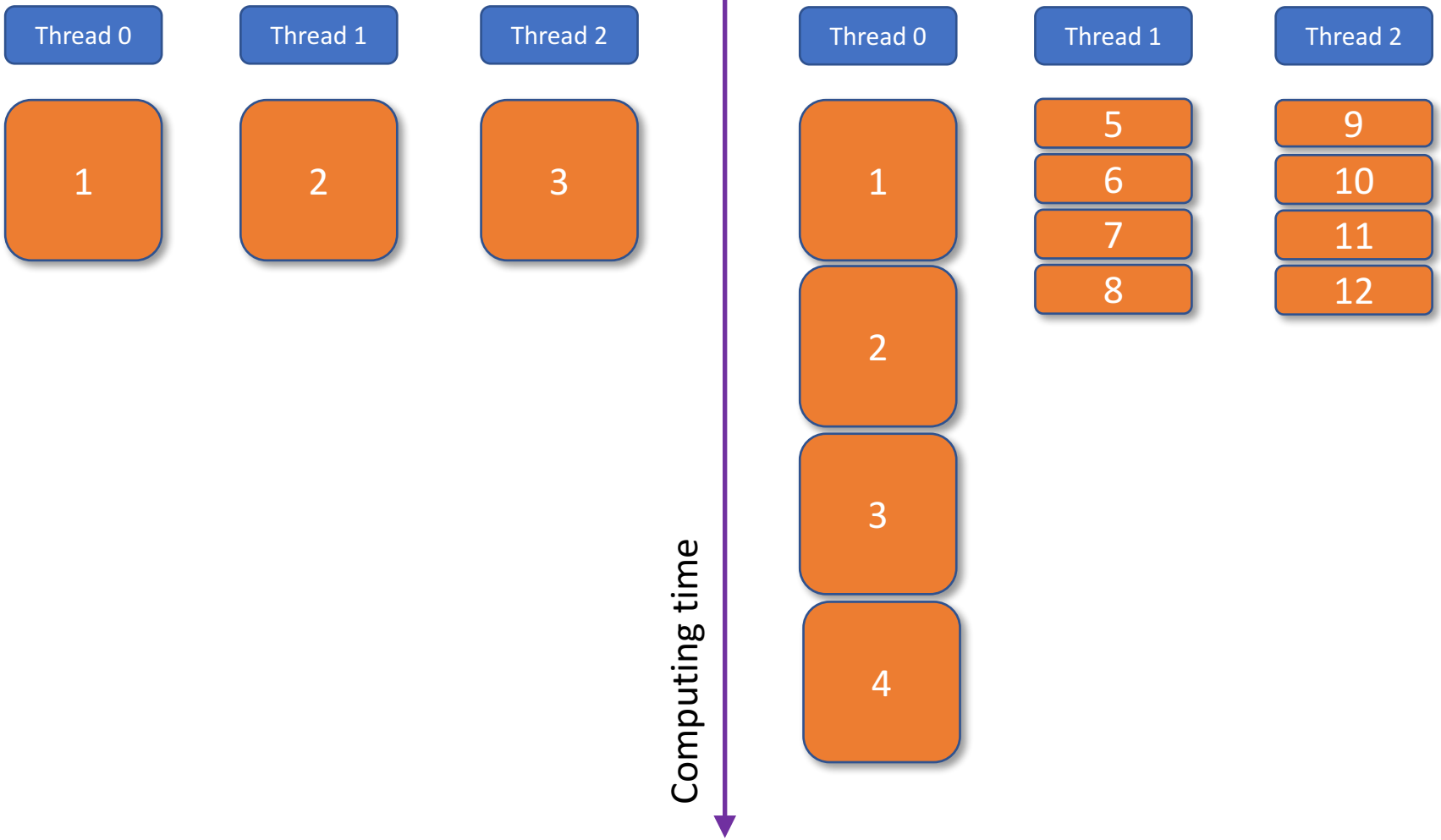

BIG ORANGE BIG IDEAS

Static schedule – fairness in terms of iterations



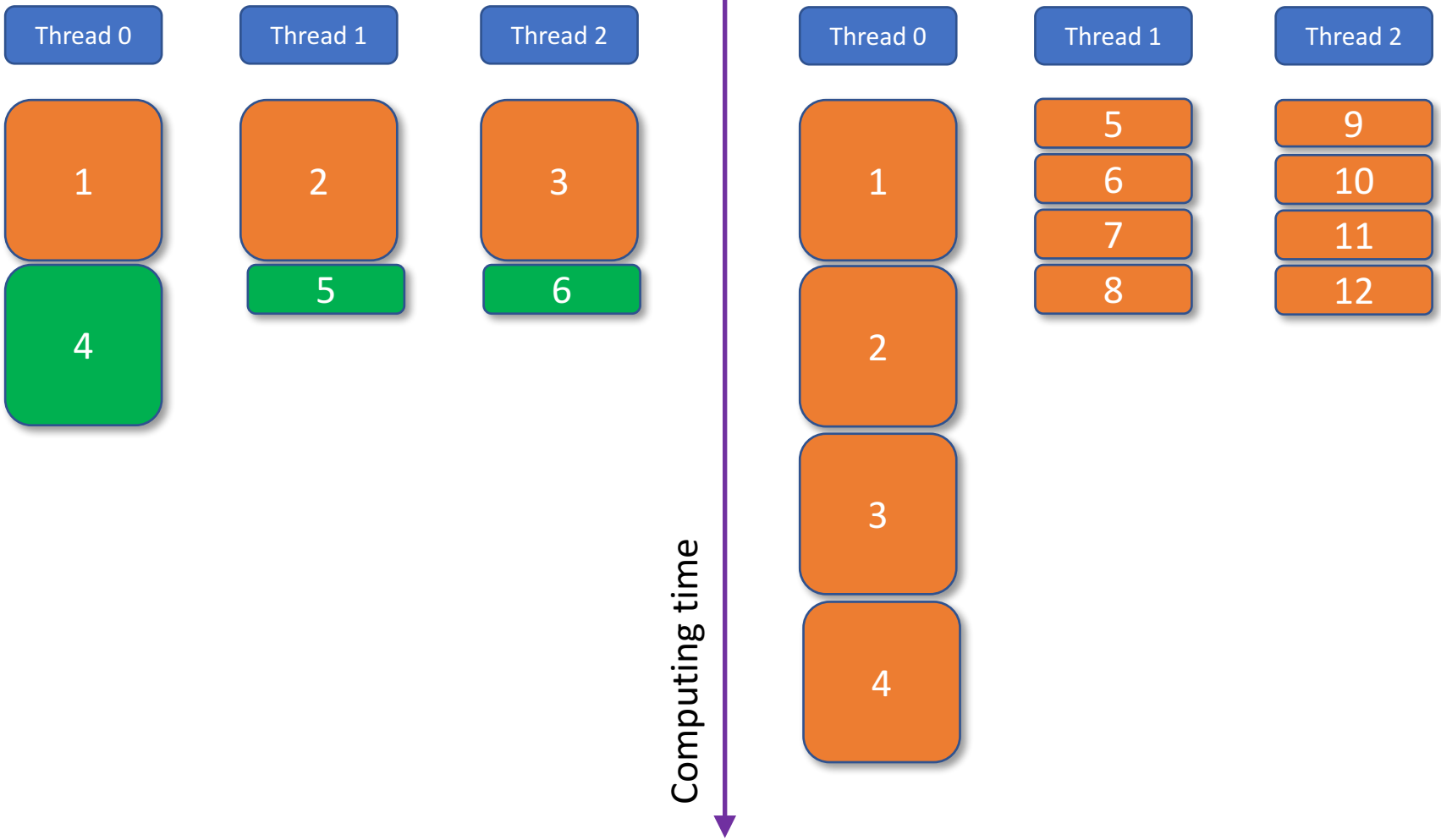
Dynamic schedule – fairness in terms of load

export OMP_SCHEDULE="dynamic,1"



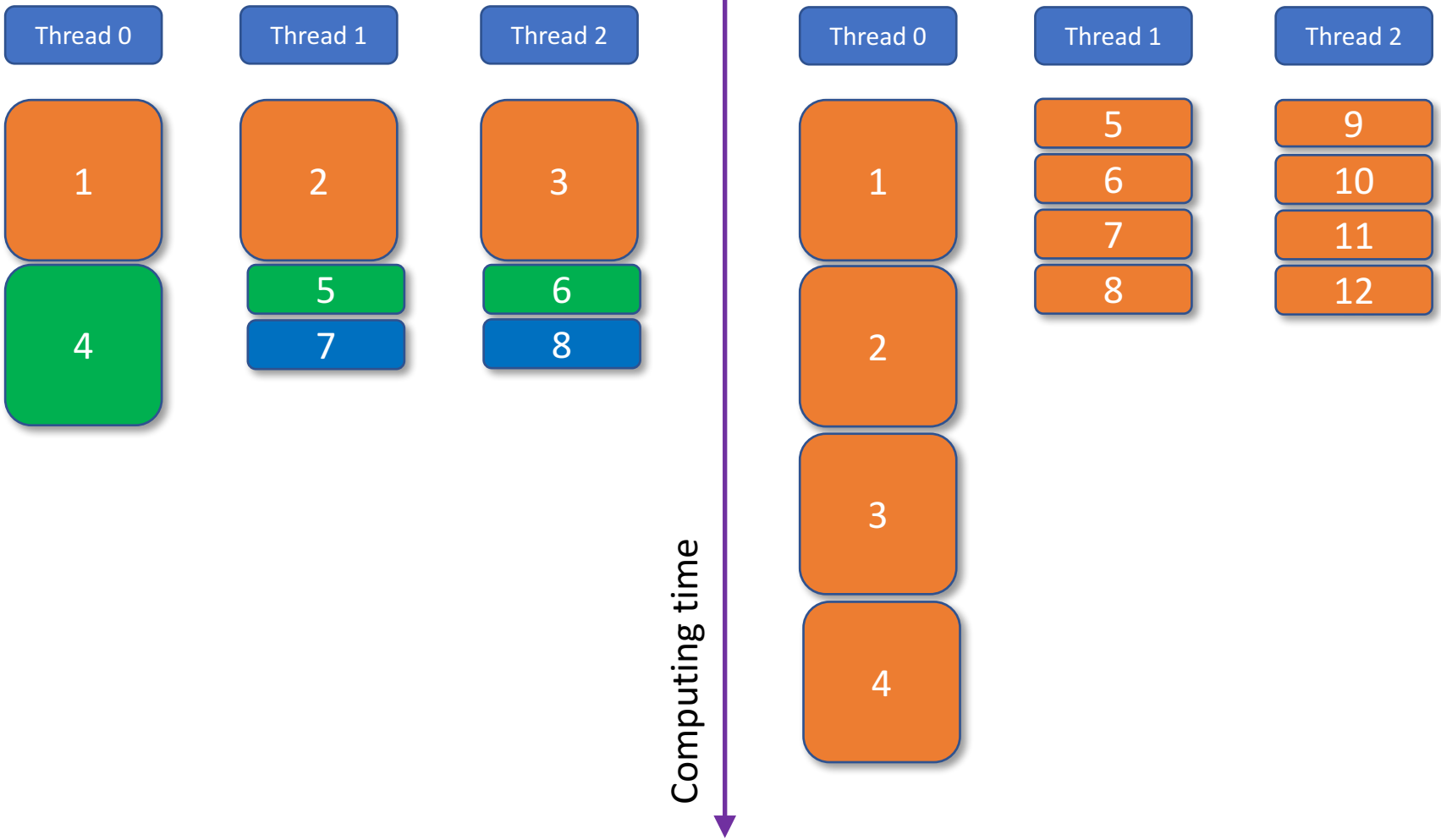
Dynamic schedule – fairness in terms of load

export OMP_SCHEDULE="dynamic,1"



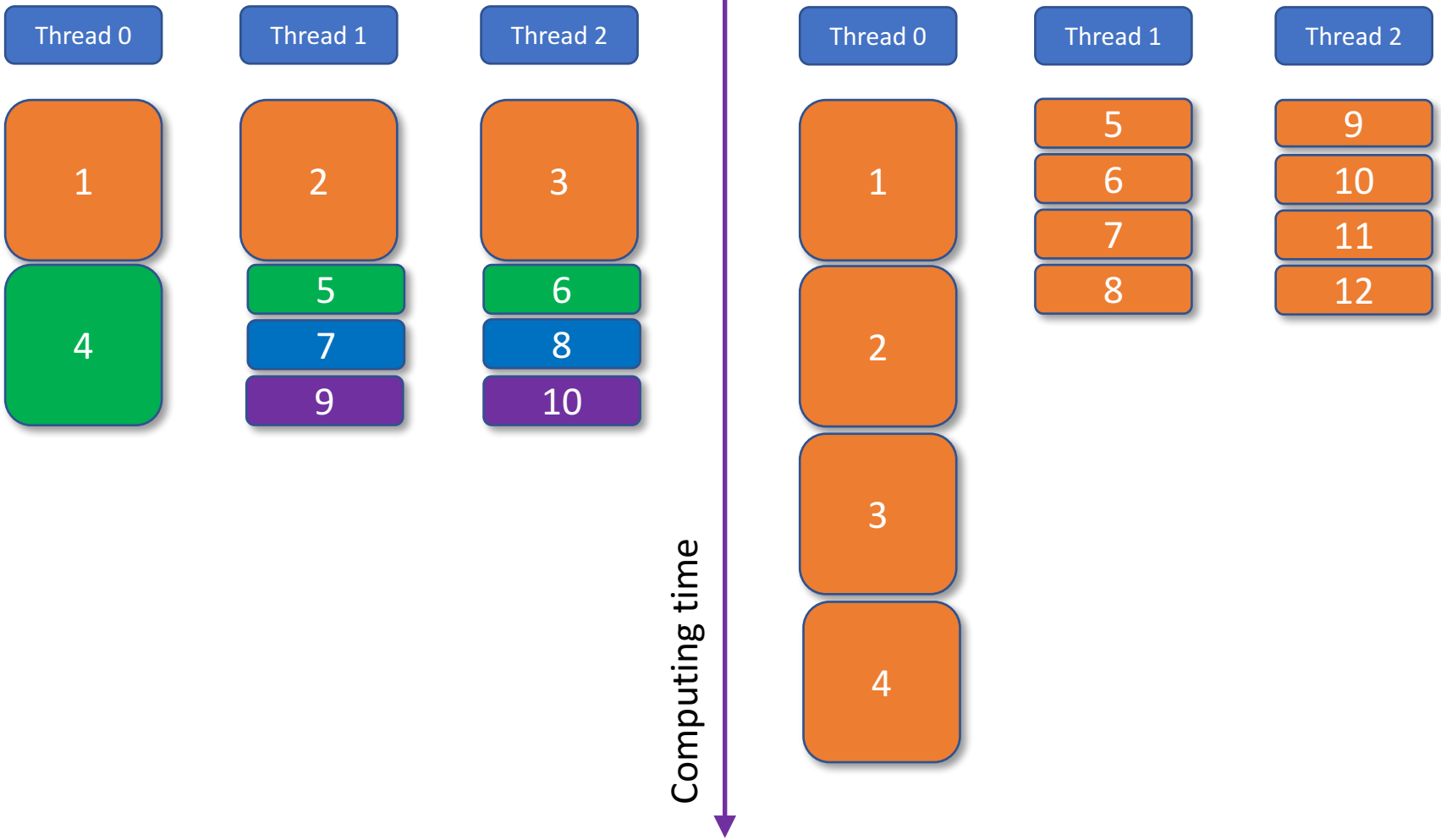
Dynamic schedule – fairness in terms of load

export OMP_SCHEDULE="dynamic,1"



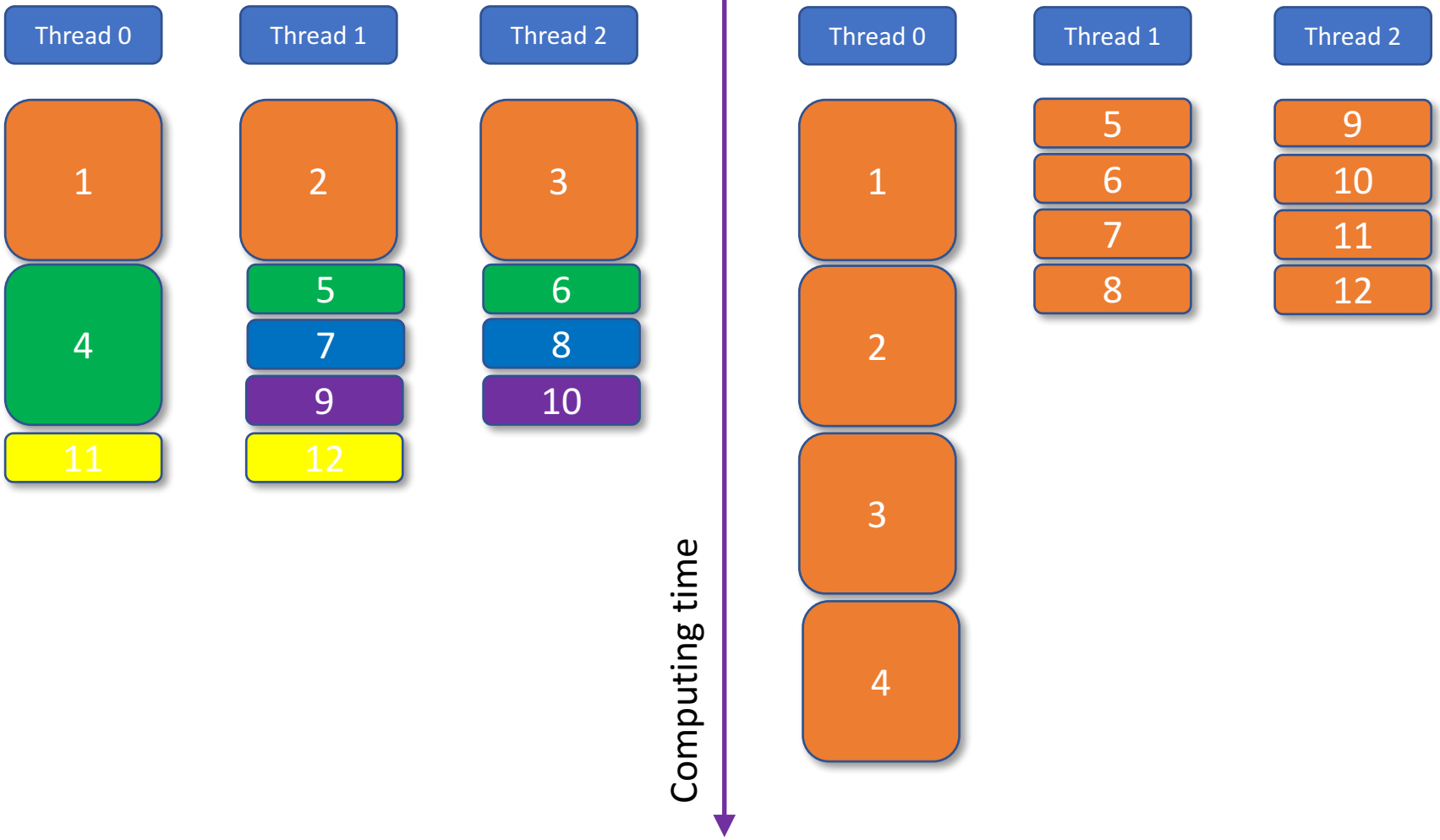
Dynamic schedule – fairness in terms of load

export OMP_SCHEDULE="dynamic,1"

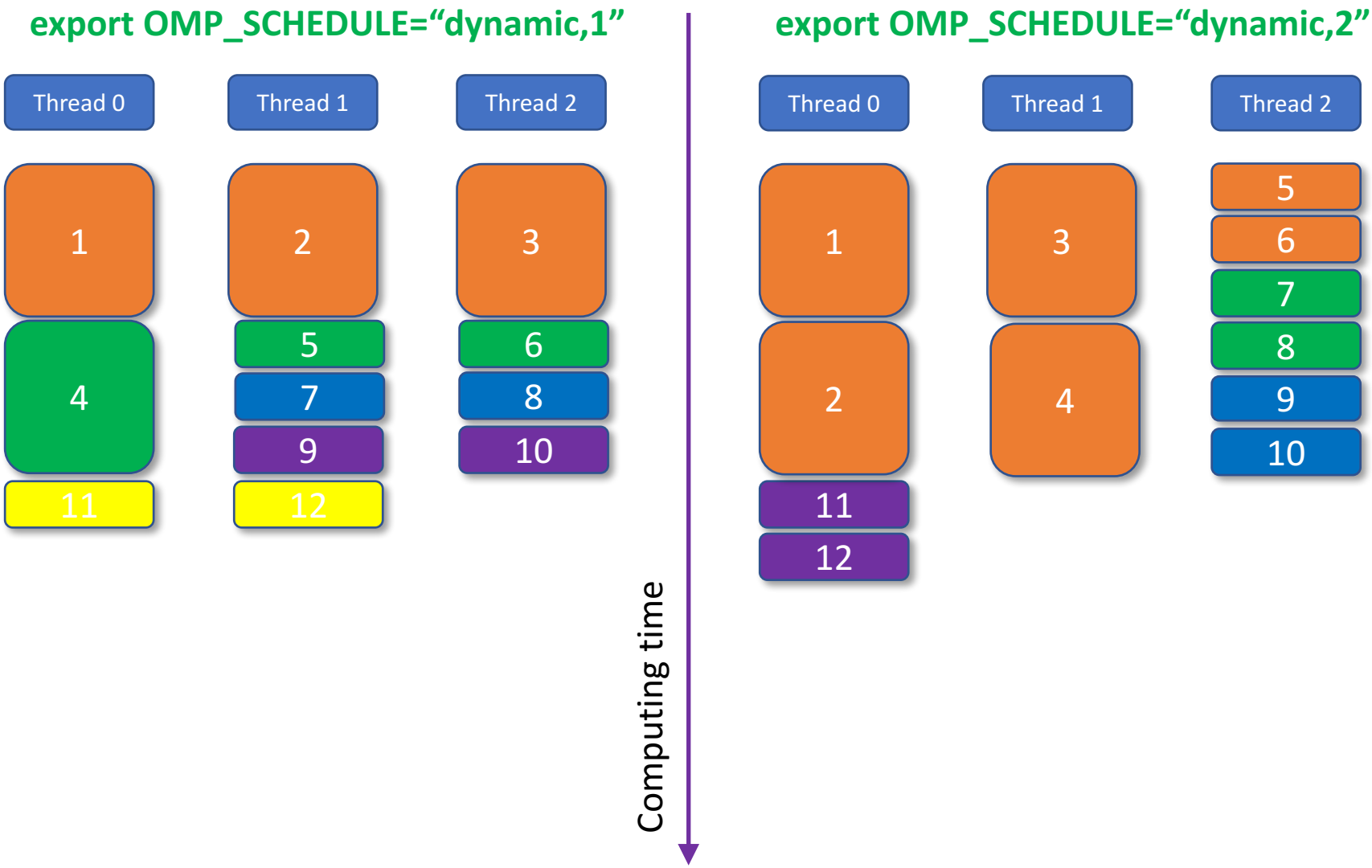


Dynamic schedule – fairness in terms of load

export OMP_SCHEDULE="dynamic,1"



Dynamic schedule – fairness in terms of load



When to use static schedule

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


int main ()
{
  int i, N;
  N = 6;


  #pragma omp parallel
  {
    #pragma omp for
    for (i = 0; i < 6; i++)
    {
      printf("Loop 1: iter %3d, thread_id %3d\n", i, omp_get_thread_num());
      do something on a[i];
      ...
      ...
    }

    #pragma omp for
    for (i = 0; i < 6; i++)
      printf("Loop 2: iter %3d, thread_id %3d\n", i, omp_get_thread_num());
      do something else on a[i];
      ...
      ...
  }

  return 0;
}
```

```
Loop 1: iter   0, thread_id   0
Loop 1: iter   1, thread_id   1
Loop 1: iter   2, thread_id   2
Loop 1: iter   3, thread_id   3
Loop 1: iter   4, thread_id   4
Loop 1: iter   5, thread_id   5
Loop 2: iter   0, thread_id   0
Loop 2: iter   1, thread_id   1
Loop 2: iter   2, thread_id   2
Loop 2: iter   3, thread_id   3
Loop 2: iter   4, thread_id   4
Loop 2: iter   5, thread_id   5
```

When to use dynamic schedule

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


int main ()
{
  int i, N;
  N = 6;


  #pragma omp parallel
  {
    #pragma omp for schedule(dynamic,1)
    for (i = 0; i < 6; i++)
    {
      printf("Loop 1: iter %3d, thread_id %3d\n", i, omp_get_thread_num());
      if (i == ...)
       {
       }
    }

    #pragma omp for schedule(dynamic,1)
    for (i = 0; i < 6; i++)
      printf("Loop 2: iter %3d, thread_id %3d\n", i, omp_get_thread_num());
       if (i == ...)
        {
        }
  }

  return 0;
}
```

Avoid parallel regions in inner loops

overheads are incurred n^2 times

```
for (i=0; i<n; i++)
for (j=0; j<n; j++)
#pragma omp parallel
#pragma omp for
for (k=0; k<n; k++)
{
 ......
}
```

Parallel construct overheads are minimized

```
#pragma omp parallel
for (i=0; i<n; i++)
for (j=0; j<n; j++)
#pragma omp for
for (k=0; k<n; k++)
{
 ......
}
```


BIG ORANGE BIG IDEAS

Collapse nested loops

```
for (i=0; i<n; i++)
for (j=0; j<n; j++)
for (k=0; k<n; k++)
{
 ......
}
```

```
#pragma omp parallel for collapse(3)
for (i=0; i<n; i++)
for (j=0; j<n; j++)
for (k=0; k<n; k++)
{
 ......
}
```

Equivalent to

```
#pragma omp parallel for
for (ijk=0; ijk<n*n*n; ijk++)
{
 ......
}
```


BIG ORANGE BIG IDEAS

Collapse nested loops

```
for (i=0; i<n; i++)
for (j=0; j<n; j++)
for (k=0; k<n; k++)
{
 ......
}
```

```
#pragma omp parallel for collapse(3)
for (i=0; i<n; i++)
for (j=0; j<n; j++)
for (k=0; k<n; k++)
{
 ......
}
```

Equivalent to

```
#pragma omp parallel for
for (ijk=0; ijk<n*n*n; ijk++)
{
 ......
}
```

```
for (i=0; i<n; i++)
x = i;
for (j=0; j<n; j++)
for (k=0; k<n; k++)
{
 ......
}
```


BIG ORANGE BIG IDEAS

Collapse nested loops

```
for (i=0; i<n; i++)
for (j=0; j<n; j++)
for (k=0; k<n; k++)
{
 ......
}
```

```
#pragma omp parallel for collapse(3)
for (i=0; i<n; i++)
for (j=0; j<n; j++)
for (k=0; k<n; k++)
{
 ......
}
```

Equivalent to

```
#pragma omp parallel for
for (ijk=0; ijk<n*n*n; ijk++)
{
 ......
}
```

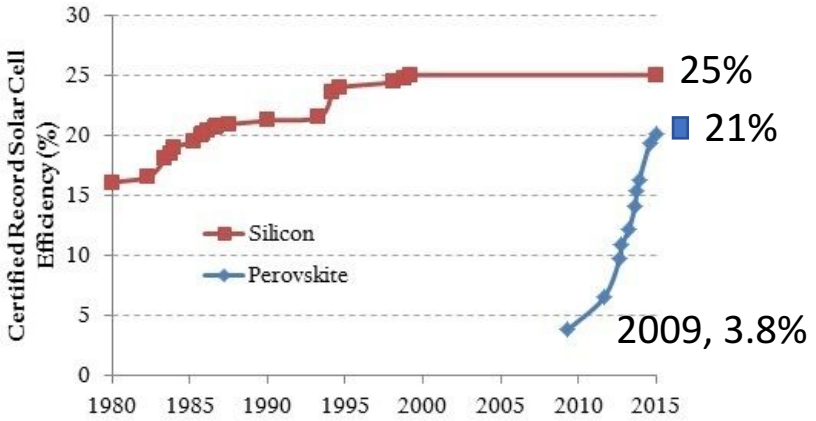
```
for (i=0; i<n; i++)
x = i;
for (j=0; j<n; j++)
for (k=0; k<n; k++)
{
 ......
}
```

```
for_nested.c(13): error: parallel loops with collapse must be perfectly nested
        x = i;
        ^
```

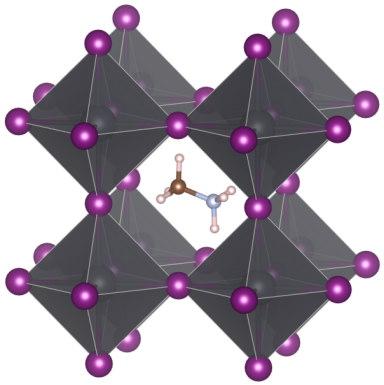

BIG ORANGE BIG IDEAS

Novel organic-inorganic perovskite AMX_3 solar cells

http://facserv.utk.edu/sustainability/on-campus/energy/



Data from National Renewable Energy Laboratory



$CH_3NH_3PbI_3$

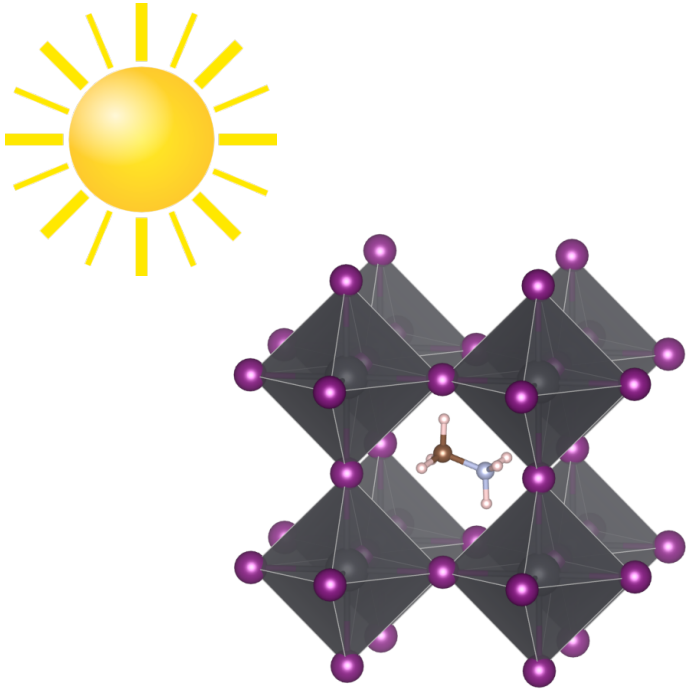
Organic | Inorganic | Perovskite

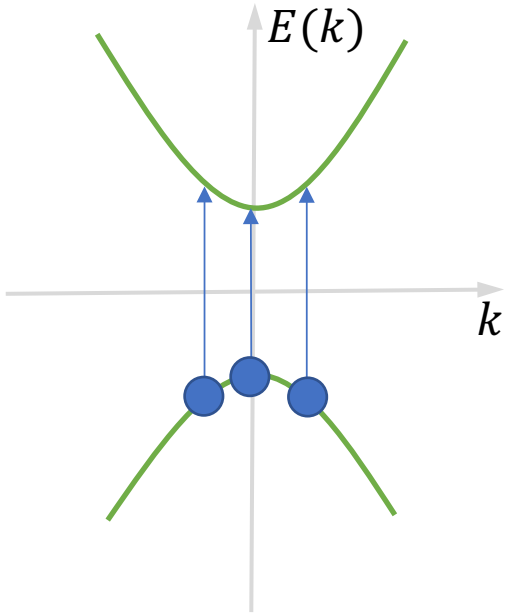
Perovskite indicates the structure of crystals like $CaTiO_3$

Perovskite materials are generally represented as AMX_3

A: Cation	M: Metal Cation	X: Anion
CH_3NH_3	Pb	I
$HC(NH_2)_2$	Sn	Cl
		Br

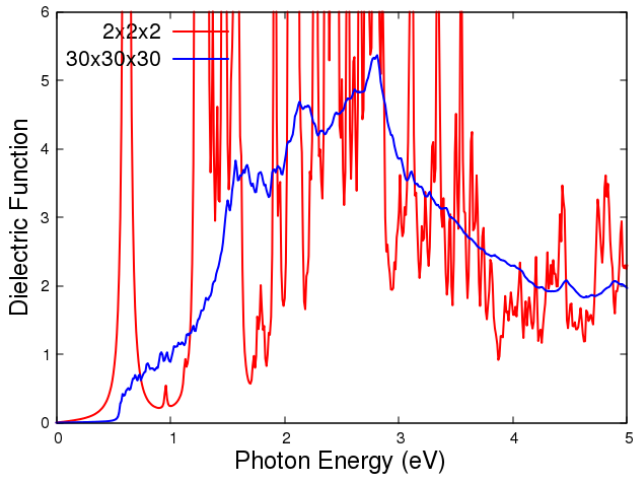
Perovskites under illumination



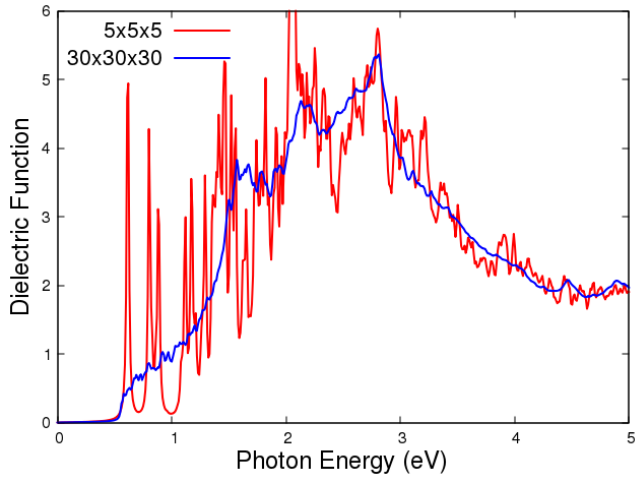


Convergence of dielectric function with respect to k-grid sampling

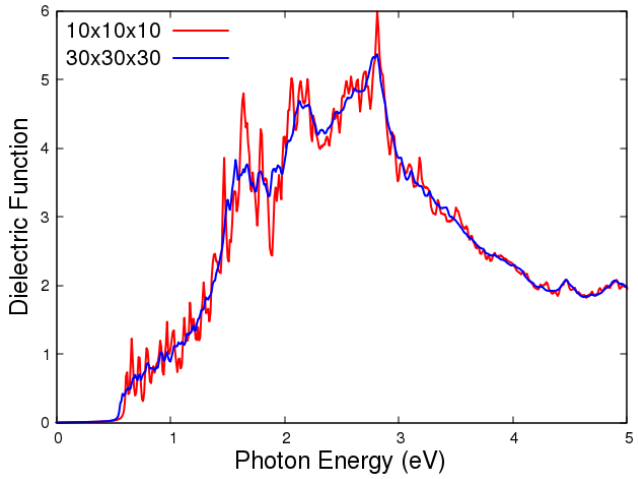
2x2x2



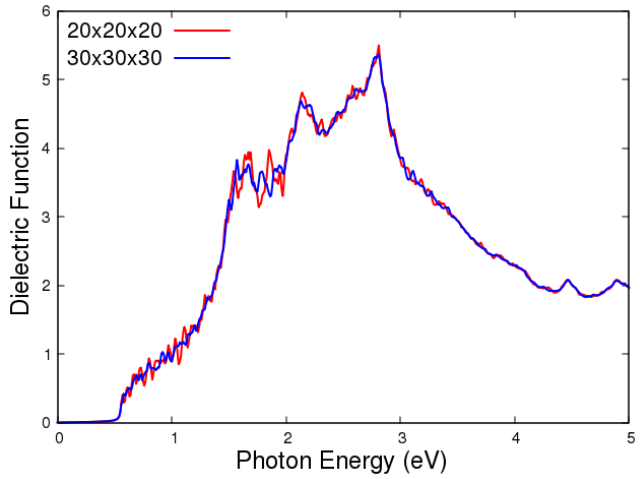
5x5x5



10x10x10



20x20x20



4.41 hours for 30x30x30

Five lines to get OpenMP parallelization of the loop over *k* points

```
use OMP_LIB
.......

block
    integer   :: ik, nbandtot, nbandval
    real (DP) :: xk(3), scissor
    nbandtot = ctrl%get_nband_tot()
    nbandval = ctrl%get_nband_val()
    scissor  = ctrl%get_scissor_shift()
    !$OMP PARALLEL PRIVATE(ik, xk, eigvec, eig, hk, dpl)
    allocate ( hk(num_wann, num_wann) )
    allocate ( eig(num_wann) )
    allocate ( eigvec(num_wann, num_wann) )
    allocate ( dpl(num_wann, num_wann, 3) )
    !$OMP DO
    do ik=1,nkpt
       xk   = kg%get_xk(ik)
       if (OMP_GET_THREAD_NUM() == 0) then
          write ( *, '(A,I15,A,I15)' ) "Working on ", ik, " / ", nkpt
       end if
       call recipe_k(p_ham,p_dpl_x,p_dpl_y,p_dpl_z,xk,num_wann,hk,eigvec,eig,dpl)
       call dist_k(wg,delta,nbandtot,nbandval,eig,scissor,dpl,epsi_k(:,:,ik),epsr_k(:,:,ik))
    end do
    !$OMP END DO
    deallocate( hk, eig, eigvec, dpl )
    !$OMP END PARALLEL
  end block

......
```

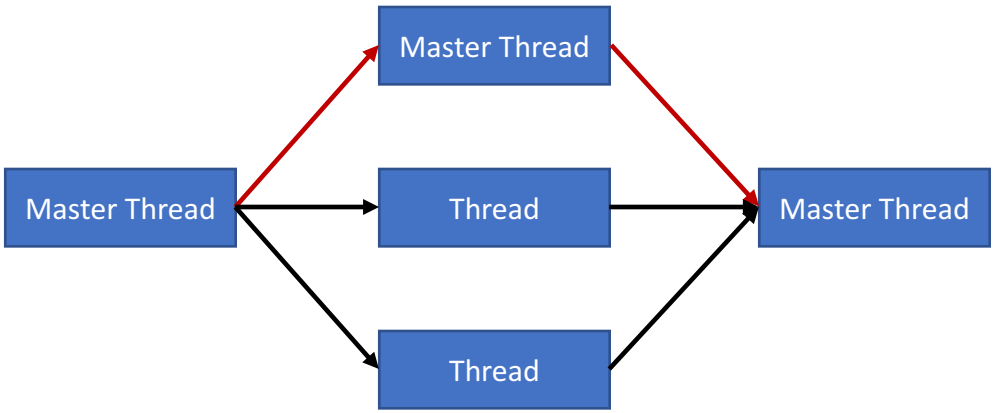

BIG ORANGE BIG IDEAS

Timing for 30x30x30

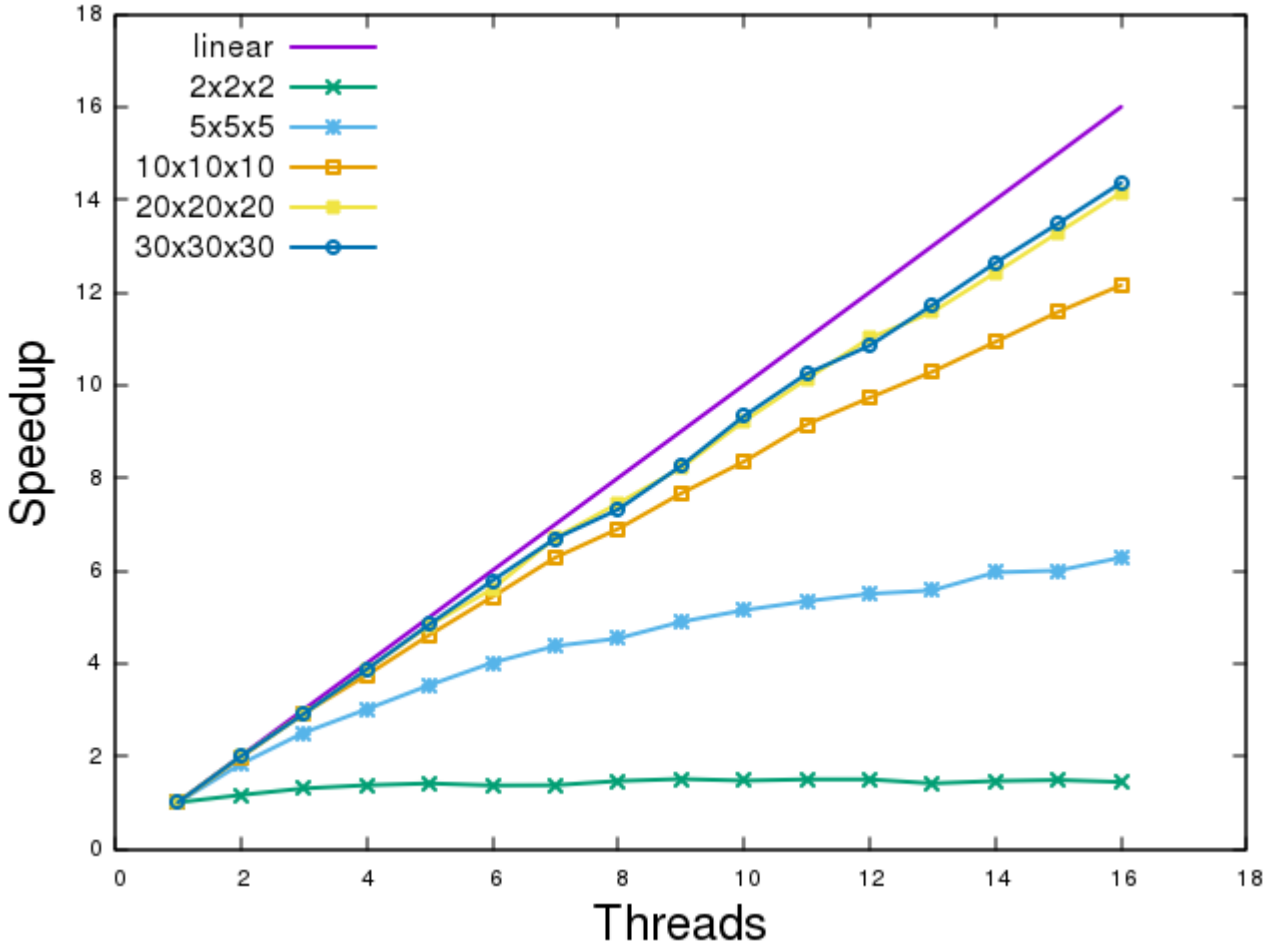
Threads	Time (seconds)	Time (minutes)	Time(hours)
1	15893.38	264.89	4.41
2	7963.32	132.72	2.21
4	4106.97	68.45	1.14
8	2171.96	36.20	0.60
16	1106.89	18.45	0.31

Demonstrate Amdahl's law by varying the number of *k* points

Load the input data Loop over *k* points Save the output data



Demonstrate Amdahl's law by varying the number of *k* points



Summary

- ❑ OpenMP syntax and directives
  - Hello world example
  - π calculation
  - Many loops or nested loops
- ❑ Demonstration with real research example
  - Five lines to parallelize the code
  - Amdahl's law

BIG ORANGE BIG IDEAS

Online resources

- **OpenMP**
  http://www.openmp.org/
- **Lawrence Livermore National Laboratory**
  https://computing.llnl.gov/tutorials/openMP
- **Dartmouth Research Computing**
  https://www.dartmouth.edu/~rc/classes/intro_openmp
- **UTexas Computing Center**
  http://pages.tacc.utexas.edu/~eijkhout/pcse/html/index.html