

```
#pragma omp parallel for num_threads(NT)
for (i=imin;i<imax;i++) coll[i-imin] = dt*(coll_term_f (i,I, F,g));
if (mpi_rank > 0) {
    MPI_Send(coll,N1,MPI_DOUBLE,0,0,MPI_COMM_WORLD)
} else {
    while (count < mpi_size) {
        MPI_Recv(tmp,N1,MPI_DOUBLE,MPI_ANY_SOURCE,0,MPI_COMM_WORLD,&mpi_status);
        sender = mpi_status.MPI_SOURCE;
        count++;
    }
}
```

Introduction to parallel programming (for physicists)

FRANÇOIS GÉLIS & GRÉGOIRE MISGUICH, IPhT courses, June 2019.



université
PARIS-SACLAY



IPhT, CEA-Saclay
Itzykson room
courses.ipht.fr

1. Introduction & hardware aspects (FG)

2. A few words about Maple & Mathematica

3. Linear algebra libraries

4. Fast Fourier transform

5. Python Multiprocessing

6. OpenMP

7. MPI (FG)

8. MPI+OpenMP (FG)

These slides (GM)

Increasing coding effort



Parallelism with Maplesoft

1. Parallel programming

[From Maple's documentation] « Maple provides tools for two different types of parallel programming. The **Task Programming Model** enables parallelism by executing multiple tasks within a single process. The second type of parallelism comes from **the Grid package**, which enables parallelism by starting multiple processes.”

Remark: the task model is somewhat analogous Threads/OpenMP, and Grid is analogous to MPI. Both OpenMP and MPI will be presented in these lectures

2. Automatic parallelization

Automatic Parallelism with

Example of automatic parallelization: simple numerical sum

- Sequential version

```
add(evalf(sin(i)), i=1..100000);
```

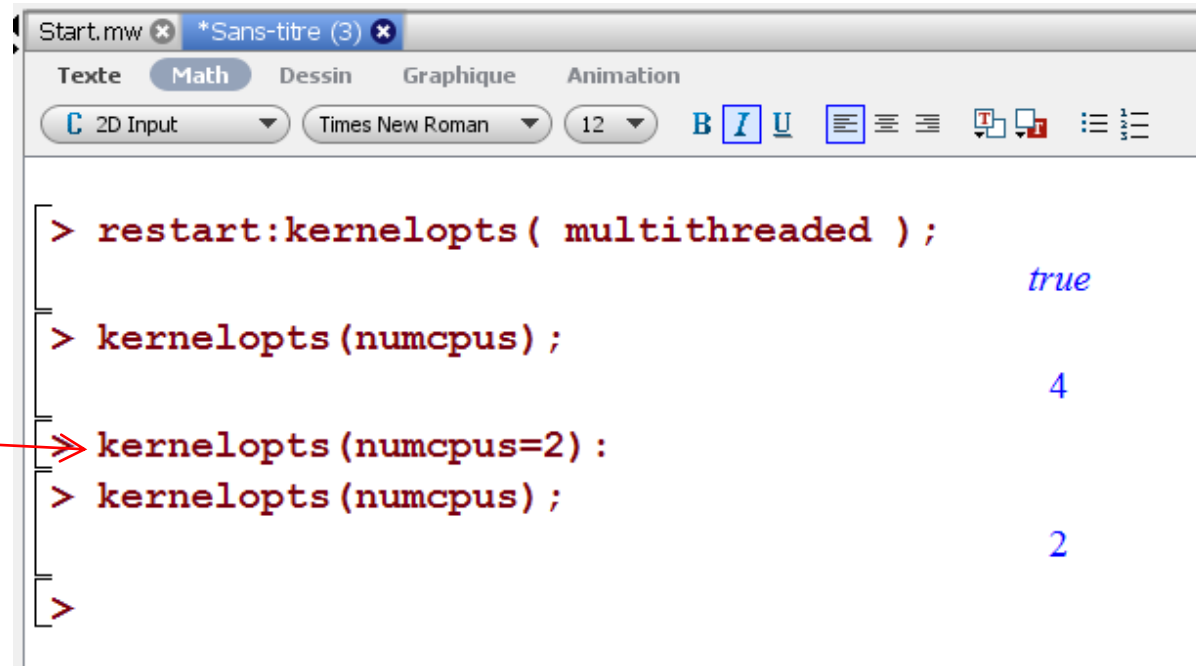
- Parallel version:

```
with(Threads);
```

```
Add(evalf(sin(i)), i=1..100000);
```

- by default Maple will create as many threads as available CPU cores
- Similar functions: **Mul**, **Seq**, and **Map**.

To select the number of CPU to be used by the multi-threaded engine:



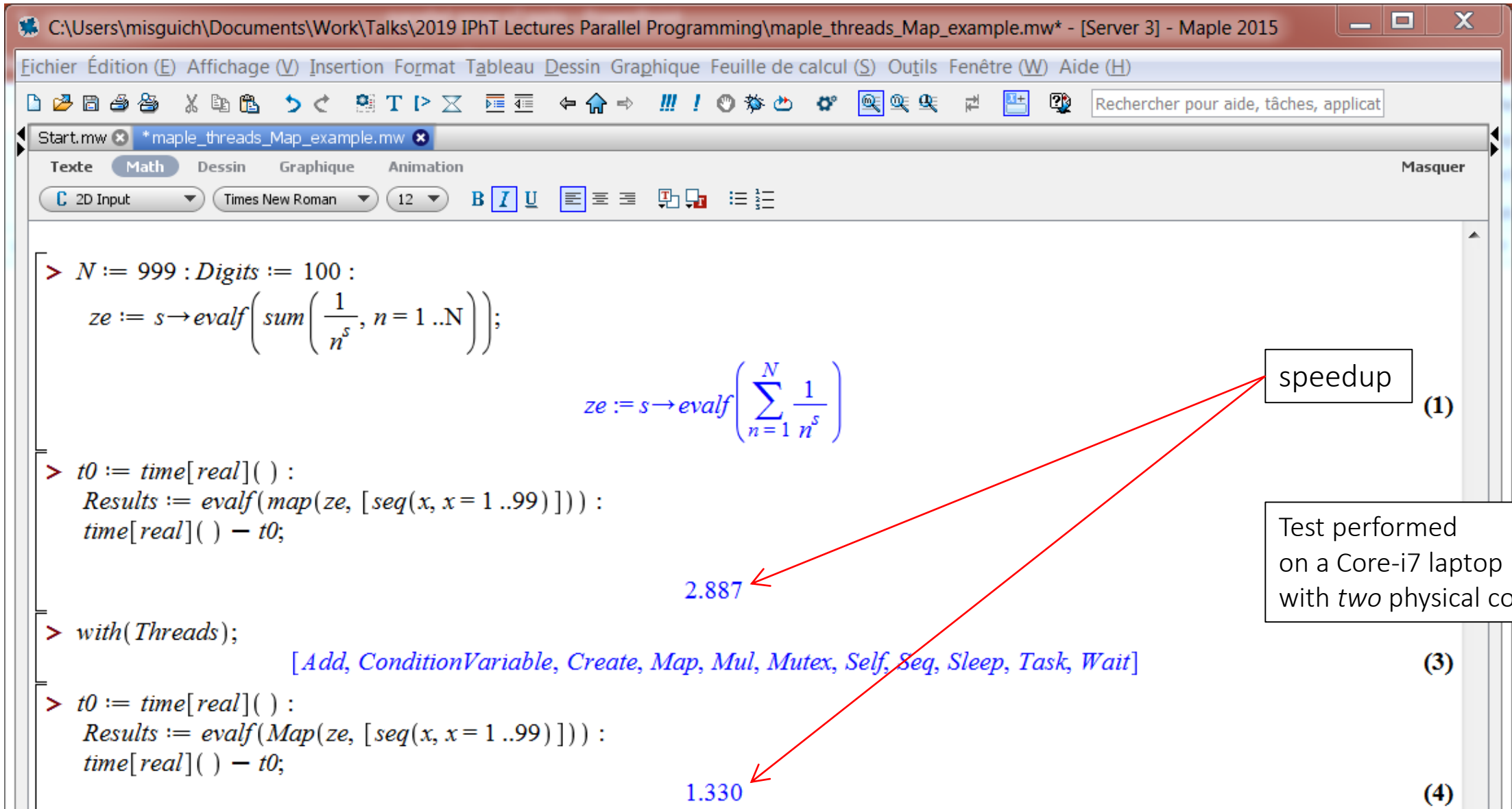
The screenshot shows the Maple software interface with a command window. The window title is 'Start.mw' and '*Sans-titre (3)'. The menu bar includes 'Texte', 'Math', 'Dessin', 'Graphique', and 'Animation'. The toolbar shows '2D Input', 'Times New Roman', '12', and various editing icons. The command window contains the following text:

```
> restart:kernelopts( multithreaded );  
true  
> kernelopts(numcpus);  
4  
=> kernelopts(numcpus=2);  
> kernelopts(numcpus);  
2  
>
```

A red arrow points from the text 'To select the number of CPU to be used by the multi-threaded engine:' to the command `kernelopts(numcpus=2);` in the screenshot.

Automatic Parallelism with

Example with `Map`



C:\Users\misguich\Documents\Work\Talks\2019 IPHT Lectures Parallel Programming\maple_threads_Map_example.mw* - [Server 3] - Maple 2015

Echier Édition (E) Affichage (V) Insertion Format Tableau Dessin Graphique Feuille de calcul (S) Outils Fenêtre (W) Aide (H)

Start.mw *maple_threads_Map_example.mw

Texte **Math** Dessin Graphique Animation Masquer

2D Input Times New Roman 12 B I U

```
> N := 999 : Digits := 100 :  
ze := s → evalf( sum( 1/n^s, n = 1 .. N ) );
```

$$ze := s \rightarrow \text{evalf} \left(\sum_{n=1}^N \frac{1}{n^s} \right)$$

```
> t0 := time[real]( ) :  
Results := evalf( map( ze, [ seq(x, x = 1 .. 99) ] ) ) :  
time[real]( ) - t0;
```

2.887

speedup (1)

```
> with( Threads );  
[Add, ConditionVariable, Create, Map, Mul, Mutex, Self, Seq, Sleep, Task, Wait]
```

Test performed on a Core-i7 laptop with two physical cores

```
> t0 := time[real]( ) :  
Results := evalf( Map( ze, [ seq(x, x = 1 .. 99) ] ) ) :  
time[real]( ) - t0;
```

1.330

(3)

(4)

Automatic parallelism with

Example with **Add**

```
> t0 := time[real]( ) :  
  add( evalf( sin( i ), i = 1 .. 100000 );  
  time[real]( ) - t0;
```

1.847777103630379156630554301110741745559264562965641397408536842921550126002349383388906799091582245

15.803

(5)

```
> t0 := time[real]( ) :  
  Add( evalf( sin( i ), i = 1 .. 100000 );  
  time[real]( ) - t0;
```

1.847777103630379156630554301110741745559264562965641397408536842921550126002349383388906799091582245

8.406

(6)

```
>
```

● Prêt

Maple Default Profile C:\Users\misguich\Documents\Work\Talks\2019 IPHT Lectures Parallel Programming Mémoire: 325.21M Temps: 95.95s Mode Math

speedup



Parallelism with Mathematica

Many possibilities:

Parallel Computing

The Wolfram Language provides a uniquely integrated and automated environment for parallel computing. With zero configuration, full interactivity, and seamless local and network operation, the symbolic character of the Wolfram Language allows immediate support of a variety of existing and new parallel programming paradigms and data-sharing models.

Automatic Parallelization

[Parallelize](#) — evaluate an expression using automatic parallelization

[ParallelTry](#) — try different computations in parallel, giving the first result obtained

[Computation Setup & Broadcasting](#) »

[ParallelEvaluate](#) — evaluate an expression on all parallel subkernels

[DistributeDefinitions](#) — distribute definitions to all parallel subkernels

[ParallelNeeds](#) — load the same package into all parallel subkernels

[Data Parallelism](#) »

[ParallelMap](#) ▪ [ParallelTable](#) ▪ [ParallelSum](#) ▪ ...

[ParallelCombine](#) — evaluate expressions in parallel and combine their results

[Concurrency Control](#) »

[ParallelSubmit](#) — submit expressions to be evaluated concurrently

[WaitAll](#) — wait for all concurrent evaluations to finish

[WaitNext](#) — wait for the next of a list of concurrent evaluations to

finish

[Shared Memory & Synchronization](#) »

[SetSharedVariable](#) — specify symbols with values to synchronize across subkernels

[SetSharedFunction](#) — specify functions whose evaluations are to be synchronized

[\\$SharedVariables](#) ▪ [\\$SharedFunctions](#) ▪ [UnsetShared](#) ▪ [CriticalSection](#)

[Setup and Configuration](#) »

[LaunchKernels](#) — launch a specified number of subkernels

[\\$KernelCount](#) — number of running subkernels

[\\$KernelID](#) ▪ [Kernels](#) ▪ [AbortKernels](#) ▪ [CloseKernels](#) ▪ ...

[\\$ProcessorCount](#) — number of processor cores on the current computer

Multi-Processor and Multicore Computation

[Compile](#) — create compiled functions that run in parallel

[Parallelization](#) — execute compiled functions in parallel

[CompilationTarget](#) — create machine-level parallel compiled functions

[GPU Computing](#) »

[CUDAFunctionLoad](#) — load a function to run on a GPU using CUDA

[OpenCLFunctionLoad](#) — load a function to run on a GPU using OpenCL

File-Based Parallelism

[FileSystemScan](#) ▪ [FileSystemMap](#)

(automatic) Parallelism with Mathematica - **Parallelize**

```
Mathematica 10.4.1 for Linux x86 (64-bit)  
Copyright 1988-2016 Wolfram Research, Inc.
```

```
In[1]:= $KernelCount  
Out[1]= 0
```

gives the number of
subkernels available for
parallel computations

```
In[2]:= AbsoluteTiming[Table[Length[FactorInteger[10^50 + n]], {n, 20}]]  
Out[2]= {2.59524, {7, 4, 6, 4, 6, 4, 3, 7, 3, 6, 6, 7, 2, 5, 3, 3, 8, 3, 7, 6}}
```

```
In[3]:= $KernelCount  
Out[3]= 0
```

```
In[4]:= AbsoluteTiming[Parallelize[Table[Length[FactorInteger[10^50 + n]], {n,  
20}]]]  
Launching kernels...  
Out[4]= {3.23171, {7, 4, 6, 4, 6, 4, 3, 7, 3, 6, 6, 7, 2, 5, 3, 3, 8, 3, 7, 6}}
```

```
In[5]:= $KernelCount  
Out[5]= 2
```

Test done on a 2-(physical)
core processor

speedup

```
In[6]:= AbsoluteTiming[Parallelize[Table[Length[FactorInteger[10^50 + n]], {n,  
20}]]]  
Out[6]= {1.55382, {7, 4, 6, 4, 6, 4, 3, 7, 3, 6, 6, 7, 2, 5, 3, 3, 8, 3, 7, 6}}
```


(automatic) Parallelism with Mathematica - ParallelMap

```
Mathematica 10.3.0 for Linux x86 (64-bit)  
Copyright 1988-2015 Wolfram Research, Inc.
```

```
In[1]:= LaunchKernels[4]
```

```
Out[1]= {KernelObject[1, local], KernelObject[2, local], KernelObject[3, local],  
KernelObject[4, local]}
```

```
In[2]:= Map[(Pause[1]; f[#]) &, {a, b, c, d}] // AbsoluteTiming
```

```
Out[2]= {4.00321, {f[a], f[b], f[c], f[d]}}
```

```
In[3]:= ParallelMap[(Pause[1]; f[#]) &, {a, b, c, d}] // AbsoluteTiming
```

```
Out[3]= {1.02335, {f[a], f[b], f[c], f[d]}}
```

```
In[4]:= CloseKernels[]
```

```
Out[4]= {KernelObject[1, local, <defunct>], KernelObject[2, local, <defunct>],  
KernelObject[3, local, <defunct>], KernelObject[4, local, <defunct>]}
```

```
In[6]:= $KernelCount
```

```
Out[6]= 0
```

```
In[7]:= LaunchKernels[2]
```

```
Out[7]= {KernelObject[5, local], KernelObject[6, local]}
```

```
In[8]:= ParallelMap[(Pause[1]; f[#]) &, {a, b, c, d}] // AbsoluteTiming
```

```
Out[8]= {2.00858, {f[a], f[b], f[c], f[d]}}
```

speedup



(automatic) Parallelism with Mathematica

Licence restrictions...

At IPhT (Mathematica network licence):

```
Mathematica 10.4.1 for Linux x86 (64-bit)  
Copyright 1988-2016 Wolfram Research, Inc.
```

```
In[1] := $MaxLicenseProcesses
```

```
Out[1] = 10
```

```
In[2] := $MaxLicenseSubprocesses
```

```
Out[2] = 80
```

8 subkernels per Process

