Status of Parallelization of FORM

Takahiro Ueda (TTP KIT Karlsruhe)

Karlsruhe Institute of Technology

Computational Theoretical Particle Physics
SFB TR9

ACAT2011 @ Brunel U.

Introduction

- FORM is a program by J. Vermaseren for symbolic manipulations of expressions consisting of a huge number of terms (~TB).
  - Now open source: http://www.nikhef.nl/~form/
    documents, forum, webCVS
- ParFORM and TFORM are parallelized versions of FORM.
  - ParFORM: the Message Passing Interface (MPI).
  - TFORM: the POSIX threads (Pthreads).
- NIKHEF Amsterdam: J. Kuipers, J. Vermaseren
- TTP Karlsurhe: J. Kühn, M. Steinhauser, T. Ueda

How FORM Works

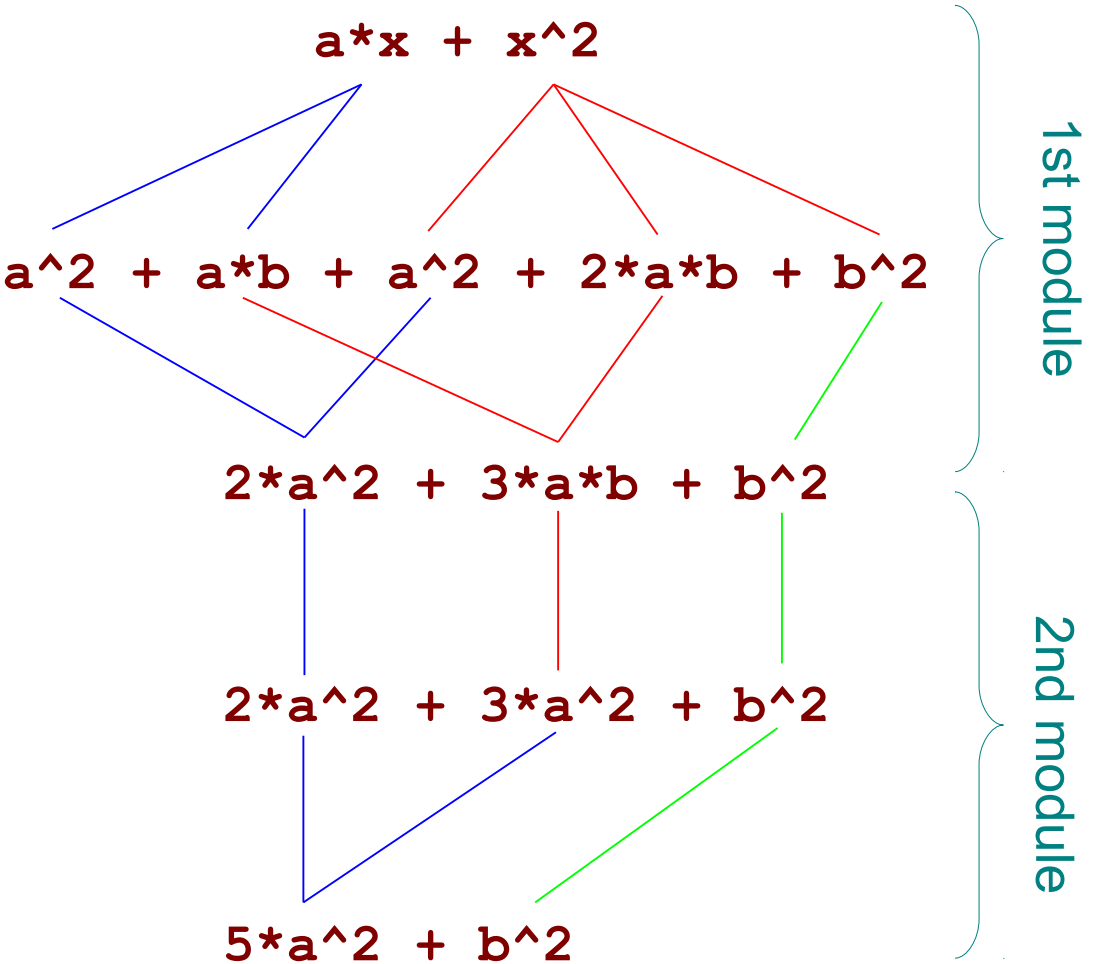
```
Symbol a,b,x;
Local expr = a*x + x^2;


id x = a + b;


.sort


if (count(b,1) == 1)
   multiply a/b;


Print;
.end
```



```
$ form example.frm
FORM by J.Vermaseren 4.0Beta(Sep  1 2011) Run at: Sun Sep  4 23:52:55 2011
    Symbol a,b,x;
    Local expr = a*x + x^2;

    id x = a + b;
    .sort

Time =       0.00 sec    Generated terms =          5
           expr          Terms in output =          3
                         Bytes used      =        108

    if (count(b,1) == 1)
        multiply a/b;

    Print;
    .end

Time =       0.00 sec    Generated terms =          3
           expr          Terms in output =          2
                         Bytes used      =         64

   expr =
      b^2 + 5*a^2;

  0.00 sec out of 0.00 sec
```

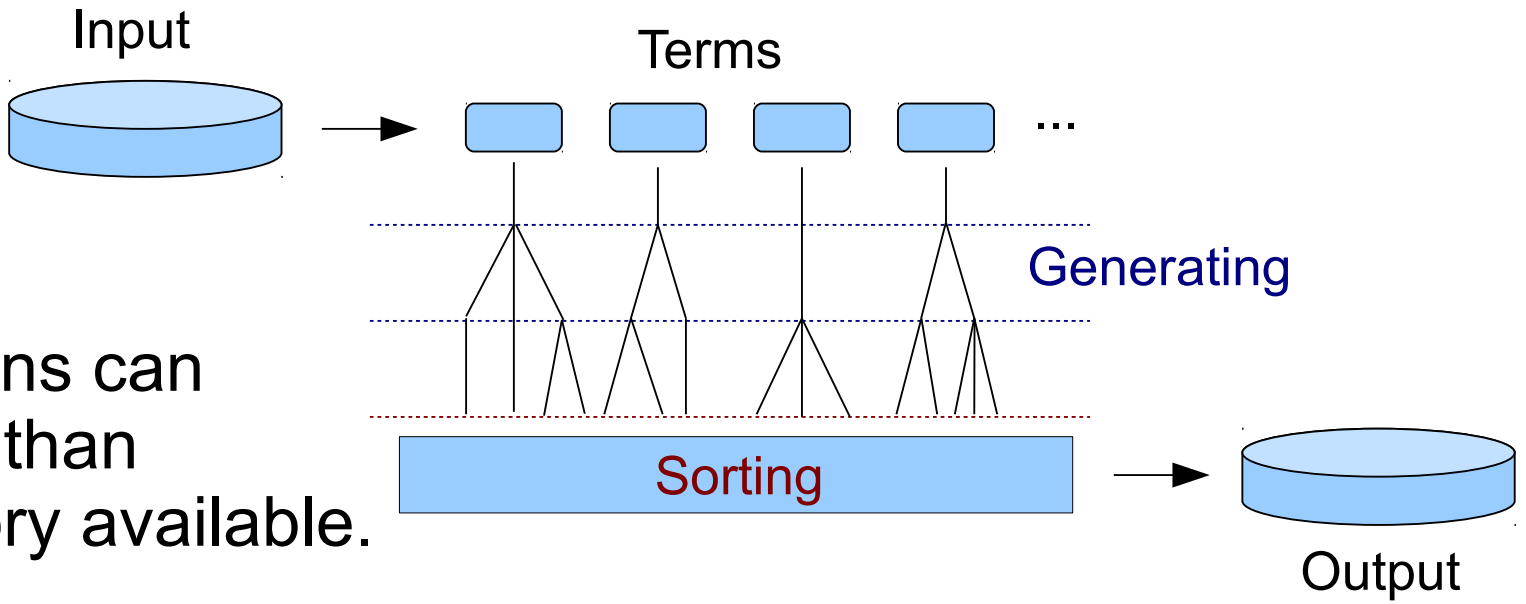
How FORM Works

- Each term in expressions is processed independently.
  - No non-local operations are allowed.
  - No common sub-expressions.

✗ `id a + b = x;`

`f(a+b)+g(2,a+b)`

- Expressions as streams of terms.
  - Sequential access to the disk storage.

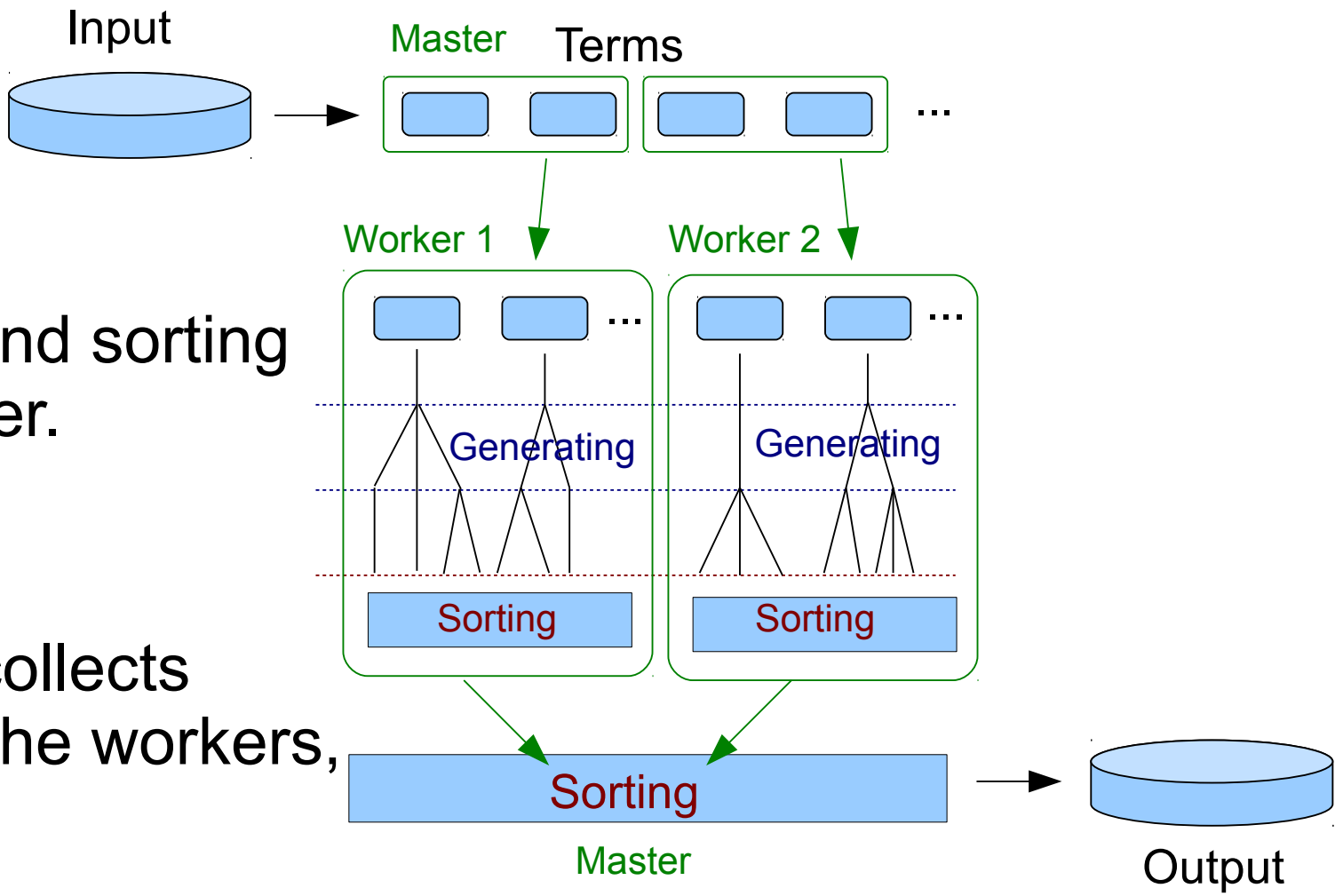


  - Expressions can be bigger than the memory available.

Of course, faster disk is better.

Concept of Parallelization of FORM

- The master distributes terms to workers (as chunks).



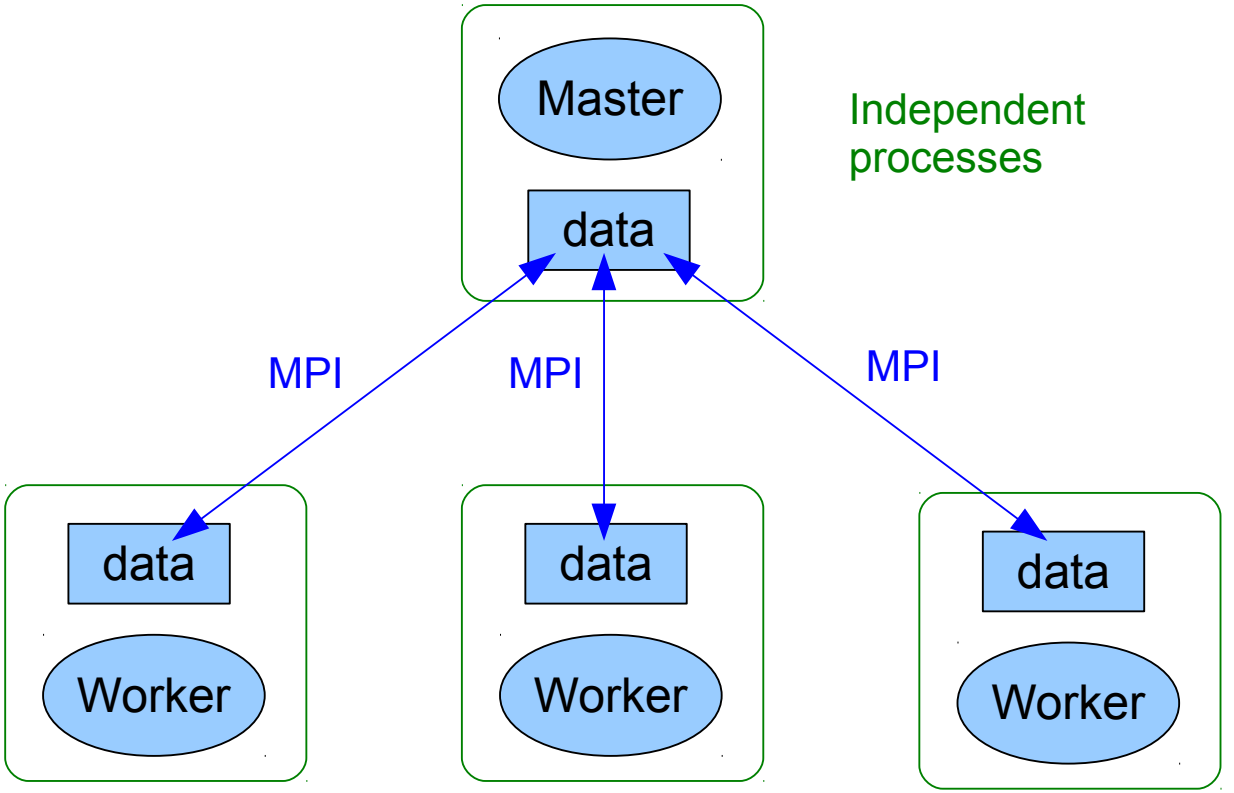
- Generating and sorting in each worker.

- The master collects results from the workers, final sorting.

ParFORM

Karlsruhe, 1998-

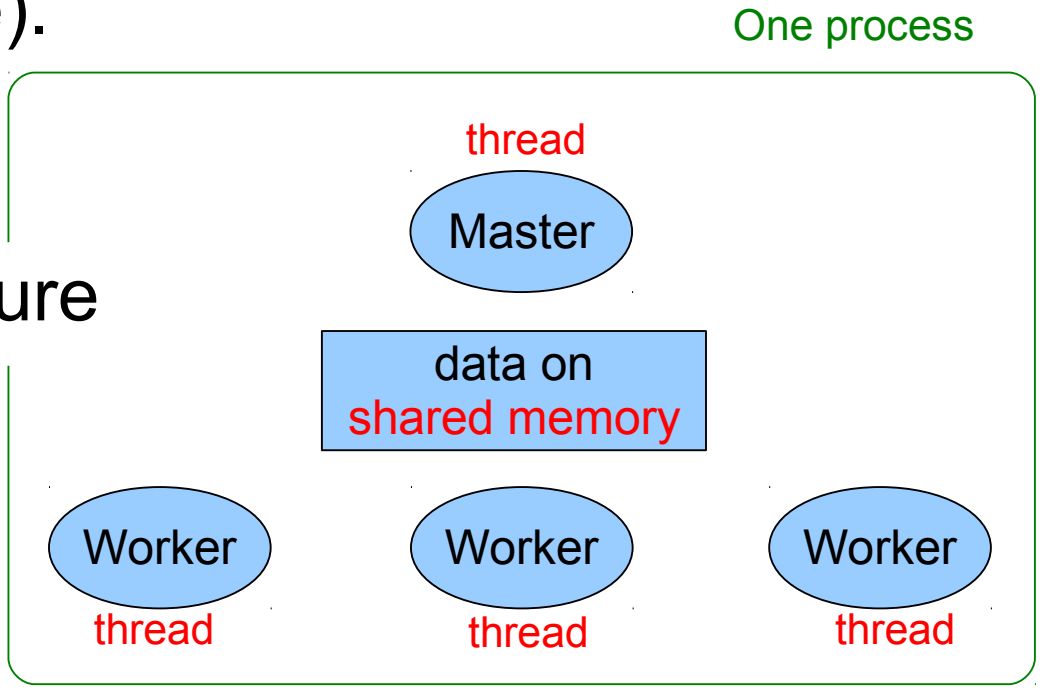
- Multiprocessor version of FORM.
- Communication via the Message Passing Interface (MPI).
- Can work on the computer cluster (w/ fast network).



TFORM

NIKHEF, 2005-

- Multithreaded version of FORM.
- Based on the POSIX threads (Pthreads).
- The master and workers share the memory.
- Performance gain on multicore processors (shared memory machine).



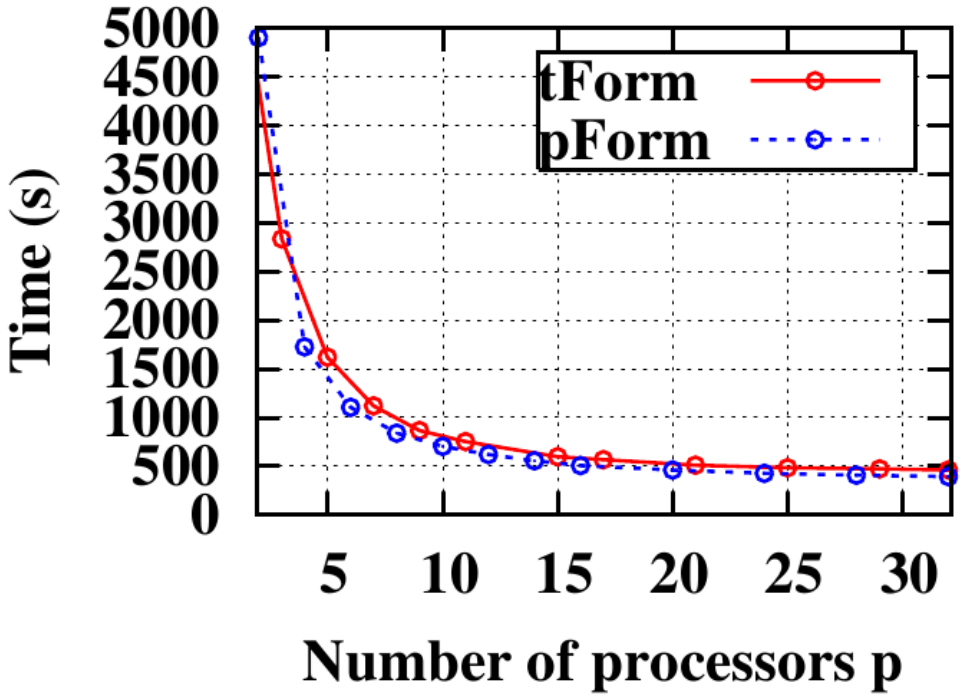
- Shared memory space allows the program structure to be much simpler than that in ParFORM.

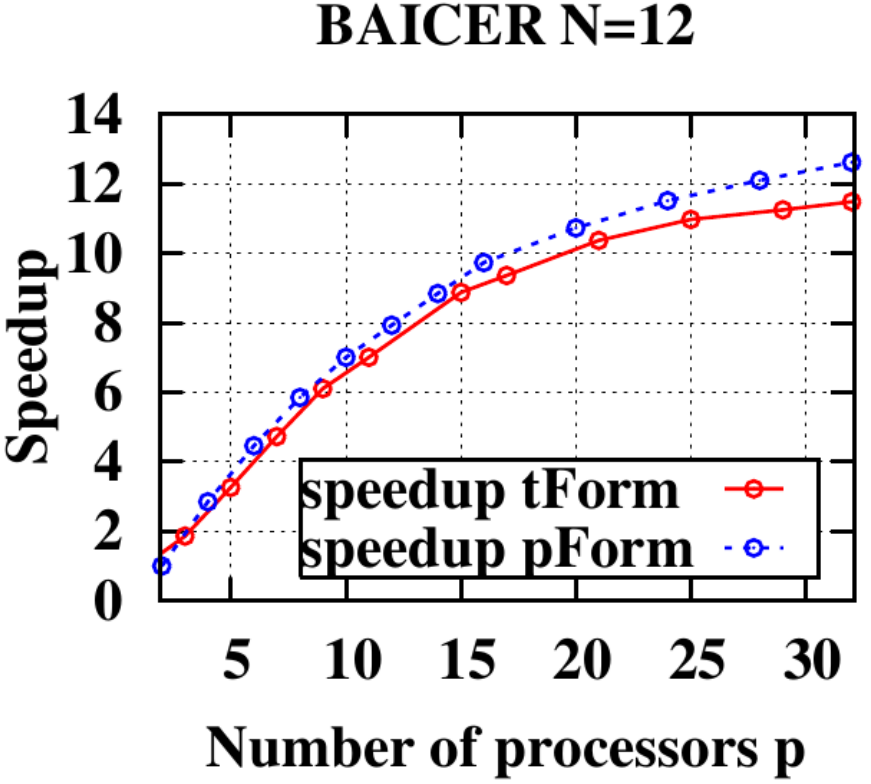
⇒ More features and tricks.

Load balancing, sortbots, ...

ParFORM vs. TFORM

- Almost same scalability:





Speedup: $S(p) = \frac{T(1)}{T(p)}$

- Parallelizations are transparent to the users: most of existing FORM programs can get a benefit without any modifications!

Parallel Versions of FORM in CVS

- ParFORM/TFORM sources are also available in the FORM CVS repository. (http://www.nikhef.nl/~form/formcvs)

```
$ wget http://goo.gl/fmyK3 -O formcvs.tar.gz
$ tar xzf formcvs.tar.gz
$ cd formcvs
$ autoreconf -i
$ ./configure --enable-parform
$ make
```

⇒ form, tform, parform in source subdirectory
(or you can “make install”)

- Run as

```
$ form myprogram.frm
$ tform -w8 myprogram.frm
$ mpirun -np 8 parform myprogram.frm
```

```
FORM by J.Vermaseren 4.0Beta(Aug 31 2011) Run at: Wed Aug 31 19:27:00 2011
    *** example of the most critical step for 4-loop integrals
    *** calculations; from ic1m10n

    #define N "12"
    #message N='N'
~~~N=12
```

```
$ form baicerN12.frm
```

```
      ⋮

Time =     2112.66 sec    Generated terms =   37500000
           F37500001 Terms left       =         120
                     Bytes used       =        9056

Time =     2113.21 sec    Generated terms =   37600000
           F37600001 Terms left       =         121
                     Bytes used       =        9124

Time =     2116.06 sec    Generated terms =   37679070
           F37679070 Terms left       =         122
                     Bytes used       =        9192

Time =     2116.06 sec
           F         Terms active     =           2
                     Bytes used       =         160

Time =     2116.06 sec    Generated terms =   37679070
           F         Terms in output  =           1
                     Bytes used       =          68

   F =
      5412793886309383467926802547115624510560000000/9*a10;

  2116.06 sec out of 2166.83 sec
```

Statistics for intermediate sorting

```
TFORM by J.Vermaseren 4.0Beta(Aug 31 2011) Run at: Fri Sep  2 12:10:43 2011
    *** example of the most critical step for 4-loop integrals
    *** calculations; from ic1m10n

    #define N "12"
    #message N='N'
~~~N=12
```

```
$ tform -w8 baicerN12.frm
```

```
      ⋮
             Thread 2 reporting
Time =     273.11 sec    Generated terms =    4757636
             F           Terms in thread =          1
                         Bytes used      =         76
             Thread 8 reporting
Time =     274.18 sec    Generated terms =    4760668
             F37664911   Terms left      =         74
                         Bytes used      =       5608
             Thread 8 reporting
Time =     274.18 sec
             F           Terms active    =          3
                         Bytes used      =        228
             Thread 8 reporting
Time =     274.18 sec    Generated terms =    4760668
             F           Terms in thread =          1
                         Bytes used      =         76

Time =     119.87 sec    Generated terms =   37679070
             F           Terms in output =          1
                         Bytes used      =         68

   F =
      5412793886309383467926802547115624510560000000/9*a10;

  119.87 sec + 2271.27 sec: 2391.15 sec out of 423.75 sec
```

Statistics for intermediate sorting on worker threads

Statistics from workers can be hidden by “Off ThreadStats;”.

```
ParFORM by J.Vermaseren 4.0Beta(Aug 31 2011) Run at: Fri Sep  2 12:17:53 2011
    *** example of the most critical step for 4-loop integrals
    *** calculations; from ic1m10n

    #define N "12"
    #message N='N'
~~~N=12
```

```
$ mpirun -np 8 parform baicerN12.frm
```

```
      ⋮
              Process 4 reporting
Time =     401.29 sec    Generated terms =    5362170
               F         Terms in process=          1
                         Bytes used      =         76
              Process 5 reporting
Time =     399.31 sec    Generated terms =    5390010
               F         Terms in process=          1
                         Bytes used      =         76
              Process 6 reporting
Time =     401.69 sec    Generated terms =    5374010
               F         Terms in process=          1
                         Bytes used      =         76
              Process 7 reporting
Time =     398.08 sec    Generated terms =    5342010
               F         Terms in process=          1
                         Bytes used      =         76

Time =     399.37 sec    Generated terms =   37679070
               F         Terms in output =          1
                         Bytes used      =         68

   F =
      5412793886309383467926802547115624510560000000/9

  399.37 sec out of 403.44 sec
```

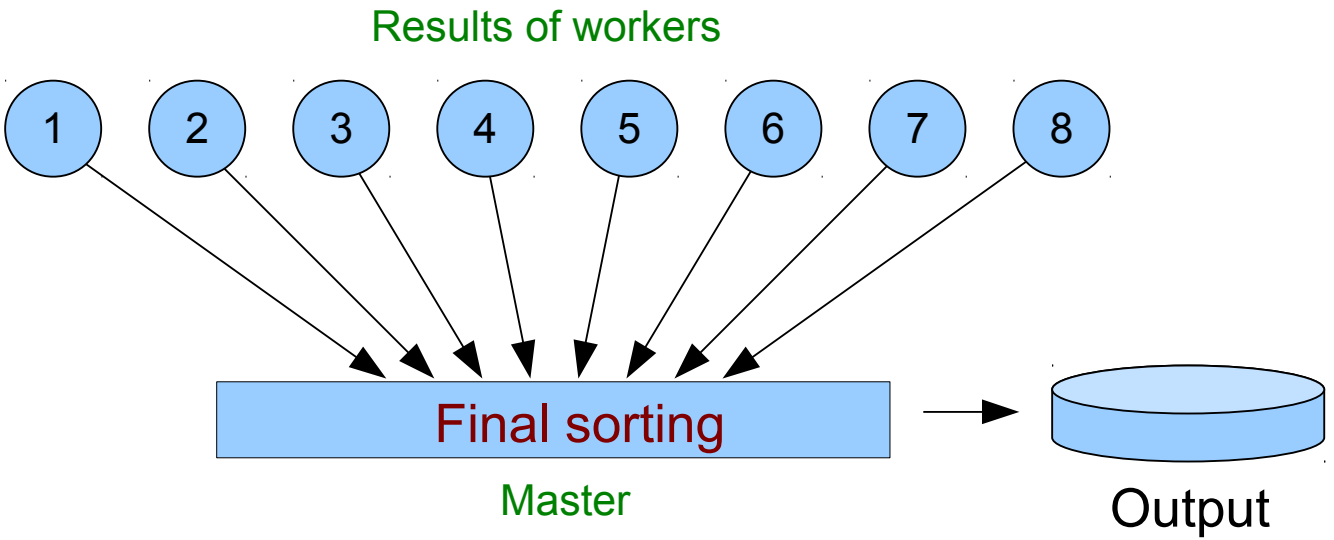
Statistics for intermediate sorting on worker processes

Statistics from workers can be hidden by “Off ProcessStats;”.

Remark for old users:
To switch back to old ParFORM statistics, “On OldParallelStats;”.

Sorting

- The final sorting is always a bottleneck.
  - The master should merge results of all workers.

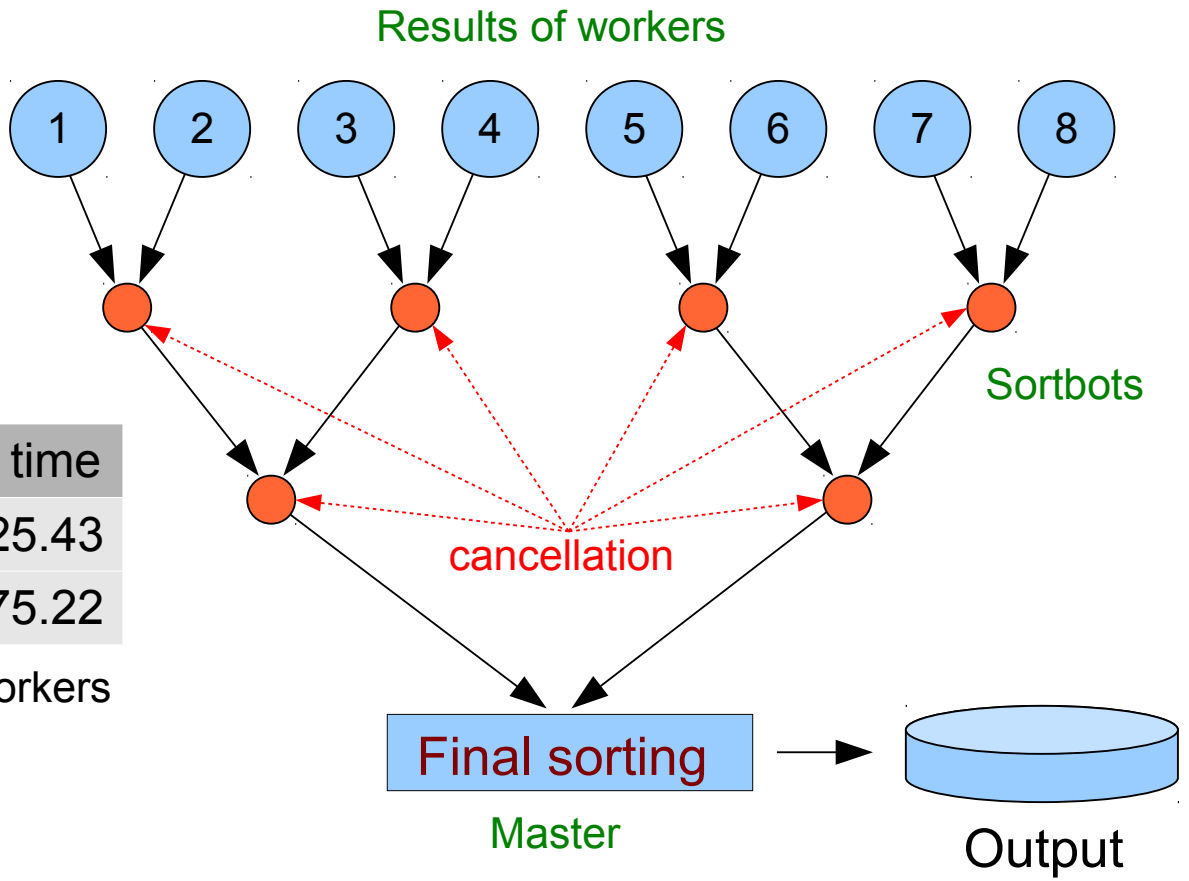


Sortbots (TFORM)

- Special threads (sortbots) merge each two results.

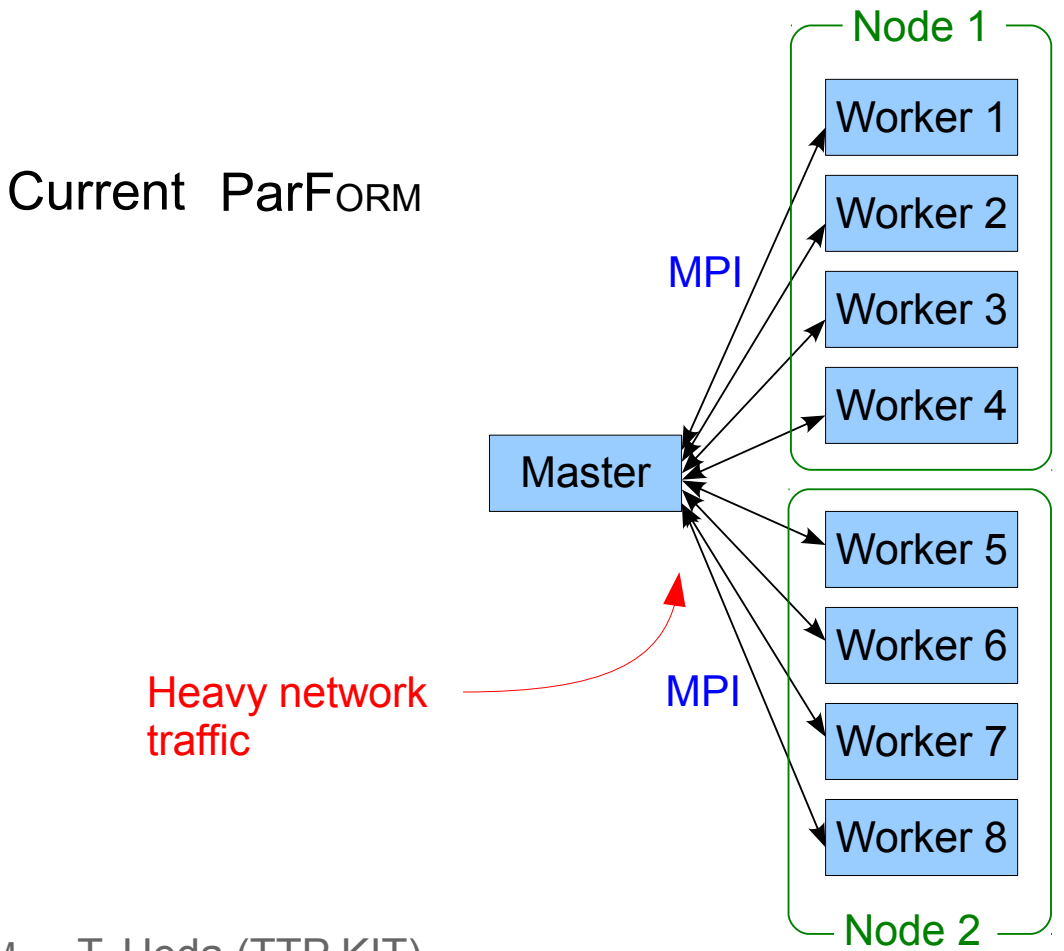
Sortbots	Master	All workers	Real time
no	125.87	1733.26	225.43
yes	62.54	1914.57	175.22

Mellin N=10, 16 workers



Future Plans of FORM Parallelization

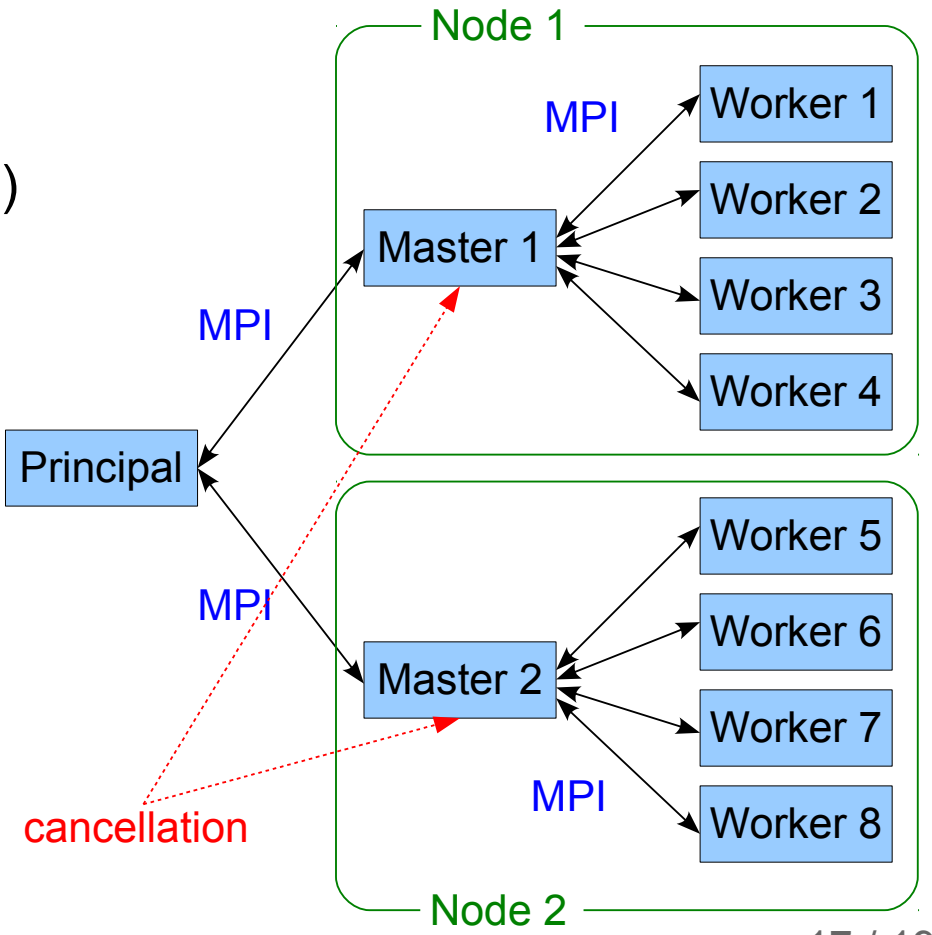
- On computer clusters built from multicore processors:
  - Heavy network traffic to the master.



Future Plans of FORM Parallelization

- On computer clusters built from multicore processors:
  - Each node has its own master.
  - Still MPI overheads in each node.

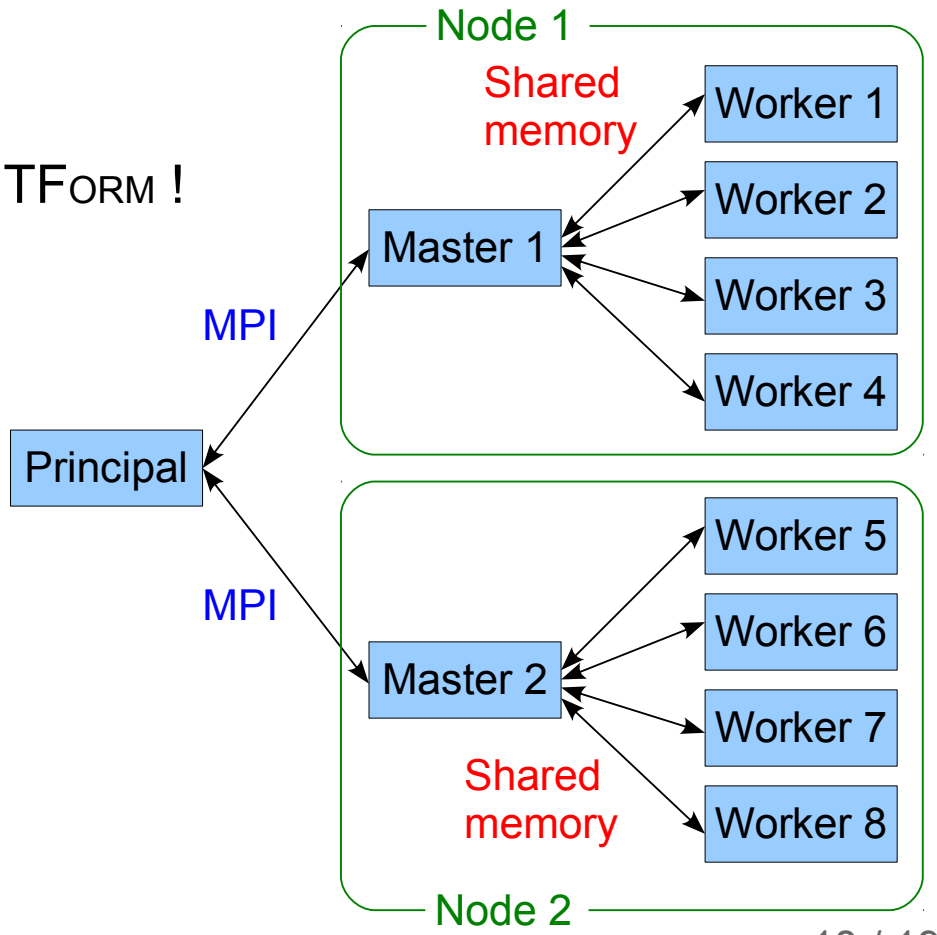
ParFORM (?)



Future Plans of FORM Parallelization

- On computer clusters built from multicore processors:
  - Hybrid MPI/Pthreads parallelization.
  - Avoid MPI overheads in each node.

ParFORM + TFORM !



Conclusion

- ParFORM and TFORM:
  - ParFORM: multiprocessor version (MPI).
  - TFORM: multithreaded version (Pthreads), using shared memory model.
- Both ParFORM and TFORM can execute almost all FORM programs in parallel.
- Both versions are in the CVS.

  http://www.nikhef.nl/~form/formcvs
- In future, ParFORM and TFORM will be combined to get advantages of each of the approaches.