ParFORM, FIRE, and FIESTA

Takahiro Ueda
TTP KIT Karlsruhe

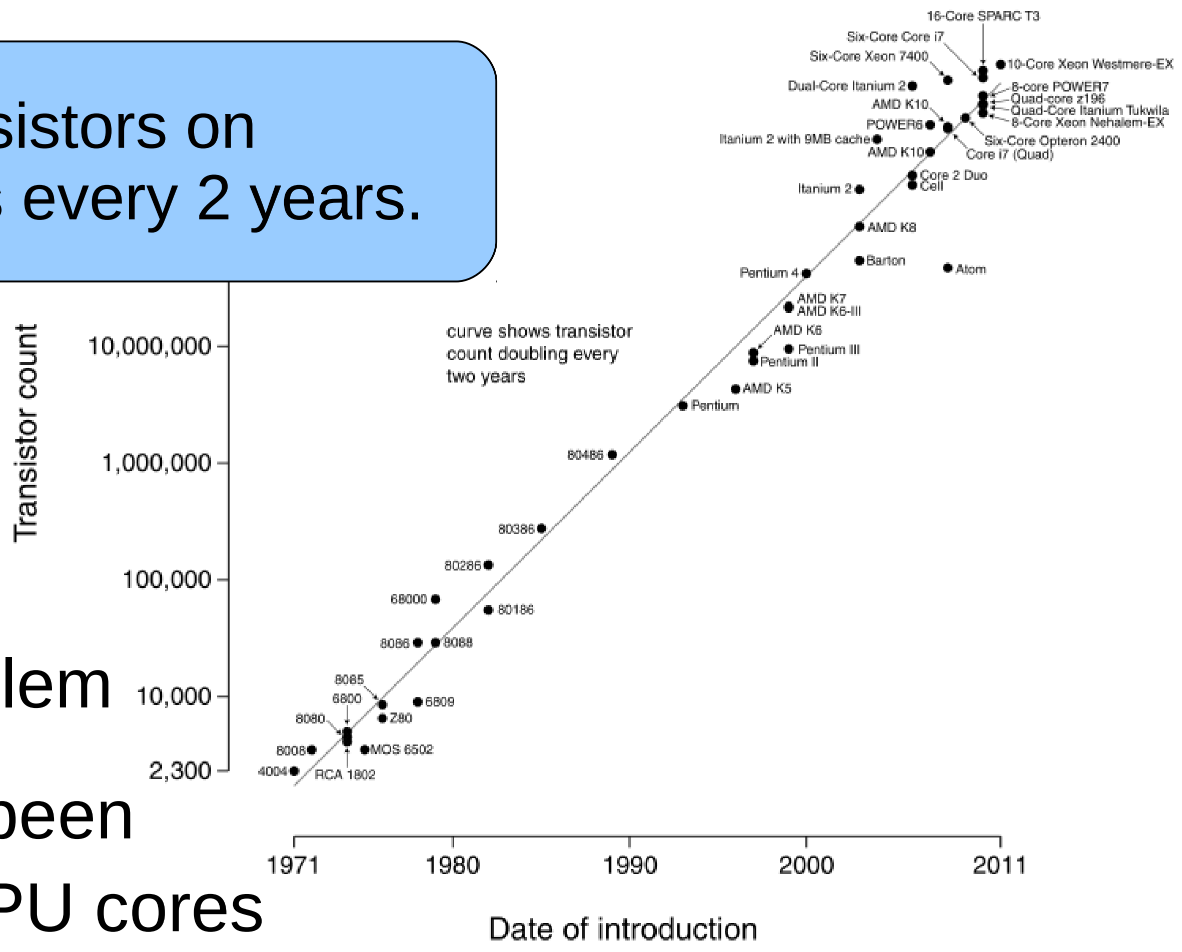
Closing Event of the SFB/TR9
Advances in Computational Particle Physics
15-19 Sep 2014, Durbach, Germany

Trends in Computing

Moore's Law

The number of transistors on
a microprocessor doubles every 2 years.

Microprocessor Transistor Counts 1971-2011 & Moore’s Law



- CPU frequency scaling until 2004 ⇨ heat problem
- Trend: transistors have been used for adding more CPU cores

⇨ Need parallelization

Mission of Project A2

- Parallelization of algebraic program systems (ParFORM, TFORM)
- Other software for Feynman integrals (FIRE, FIESTA)

  ⇨ Needed for A1 and other projects
- For example, in the typical approach for multi-loops,
  - Feynman diagram generation
  - Algebraic manipulations to scalar integrals ⇨ FORM
  - Reduction to master integrals ⇨ FIRE
  - Evaluation of master integrals ⇨ FIESTA

  or FORM for all the three steps (MINCER, MATAD, ...)

ParFORM

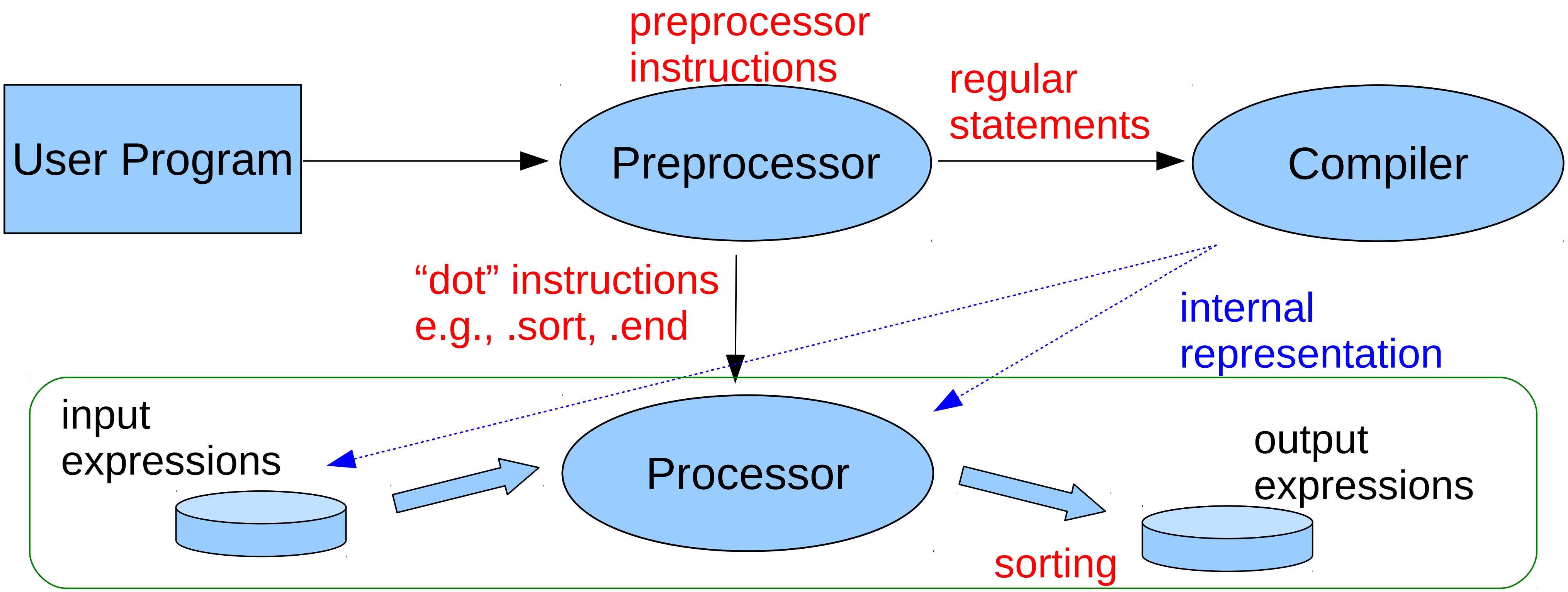
- One of parallel versions of FORM ⇨ Vermaseren's talk
  - cf. TFORM

Source code : https://github.com/vermaseren/form

- Use MPI (Message Passing Interface) for inter-process communication
- Algebraic manipulation tasks are automatically distributed over worker processes
- Parallelization is transparent for users

How FORM works

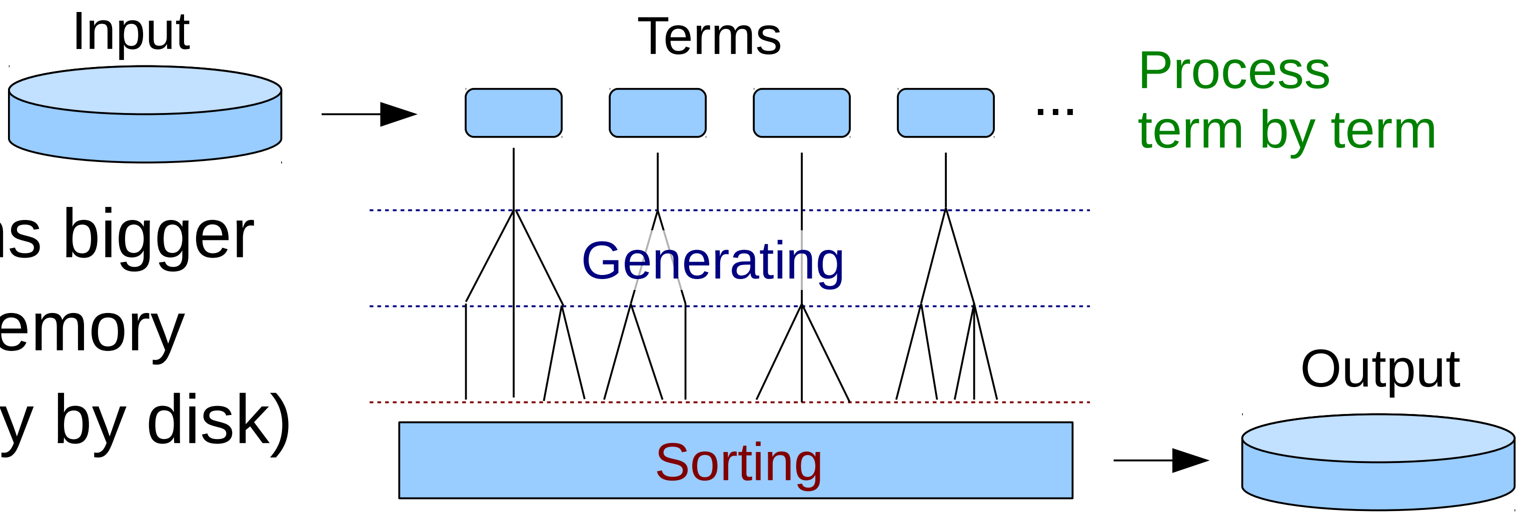
- Preprocessor: preparation and filter of the user input
- Compiler: compiles statements to internal representation
- Processor: execution of statements, generation of terms and sorting them



Sequential FORM

- Locality principle:
  - Operations are local for each term
  - Complete data are stored locally for each term

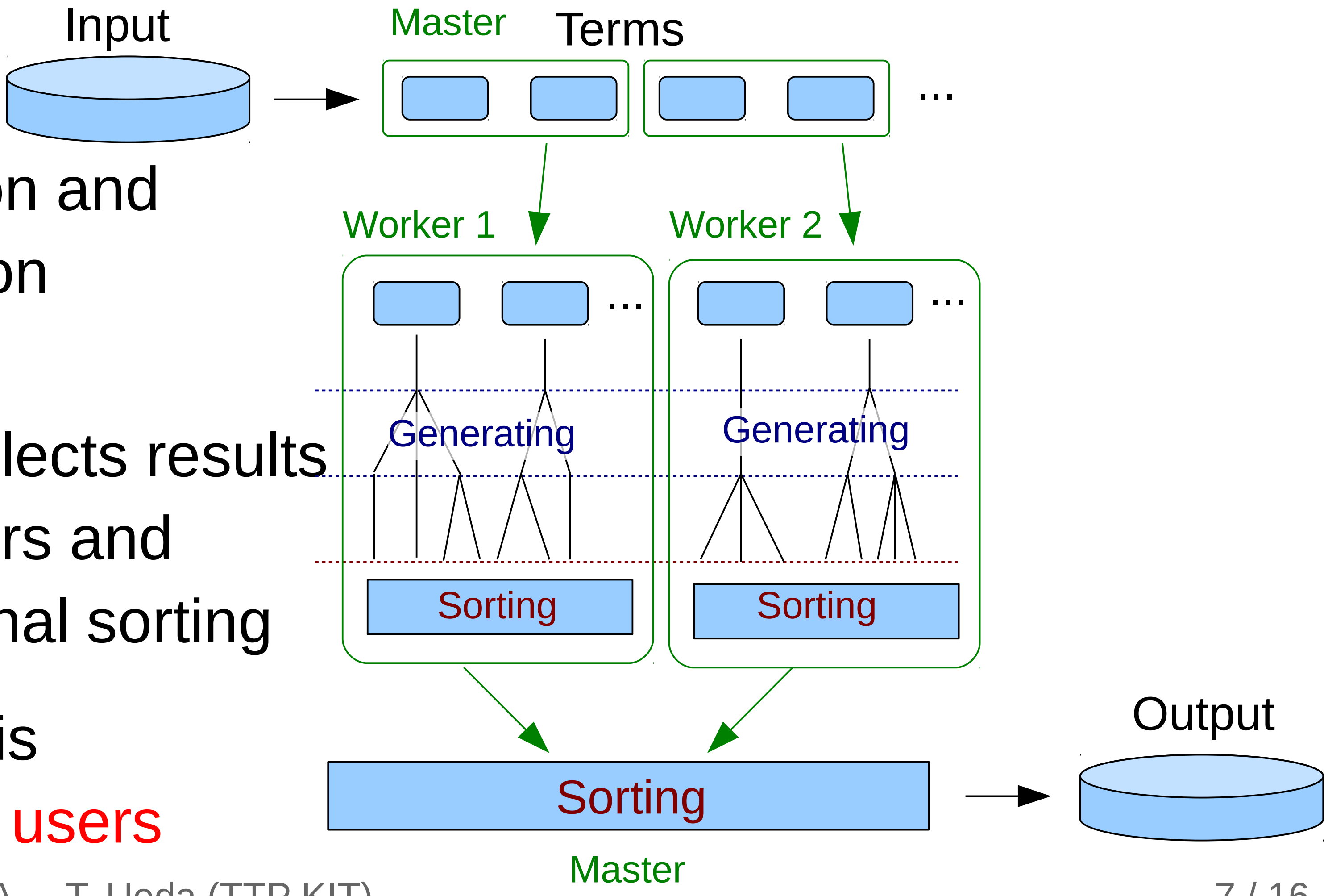
    ⇨ Can process each term independently
- Expressions as “streams” of terms
  - Sequential access to the disk storage. Merge sort on disk



  - Expressions bigger than the memory (limited only by disk)

Parallelization of FORM

- Based on master-worker model
- The master distributes terms to workers



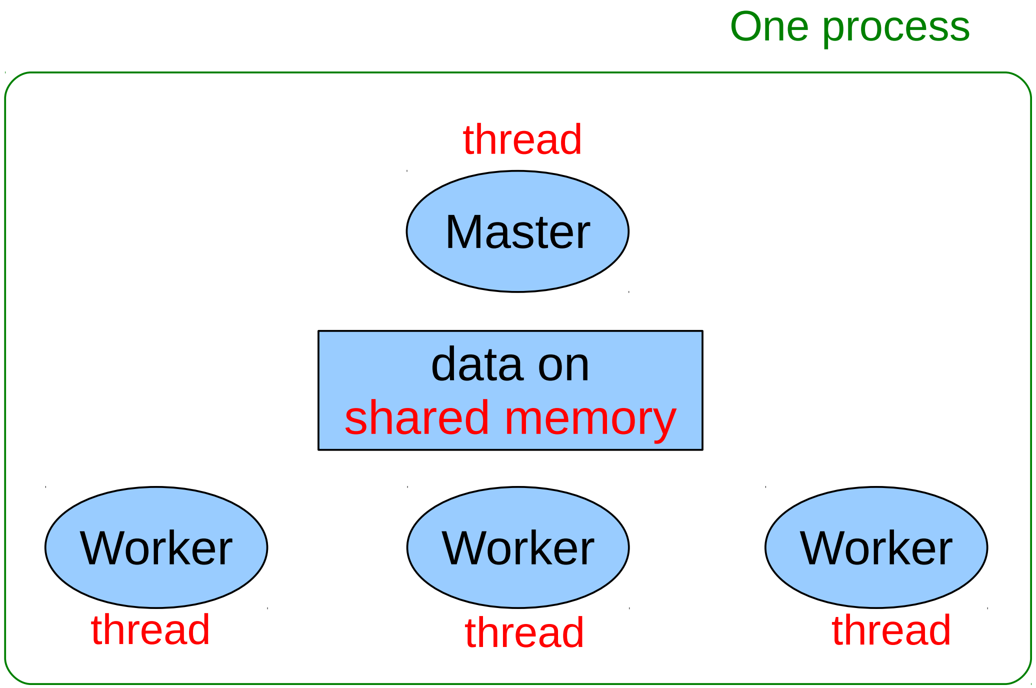
- Term generation and partial sorting on each worker
- The master collects results from the workers and performs the final sorting
- Parallelization is transparent for users

TFORM

- Multithreaded version of FORM
- Based on the POSIX threads (Pthreads)
- Communication via the shared memory space
- Performance gain on multicore computers

NIKHEF, 2005-

Tentyukov, Vermaseren ’10



ParFORM

- Multiprocessor version of FORM
- Communication via the Message Passing Interface (MPI)
- Can run on computer clusters

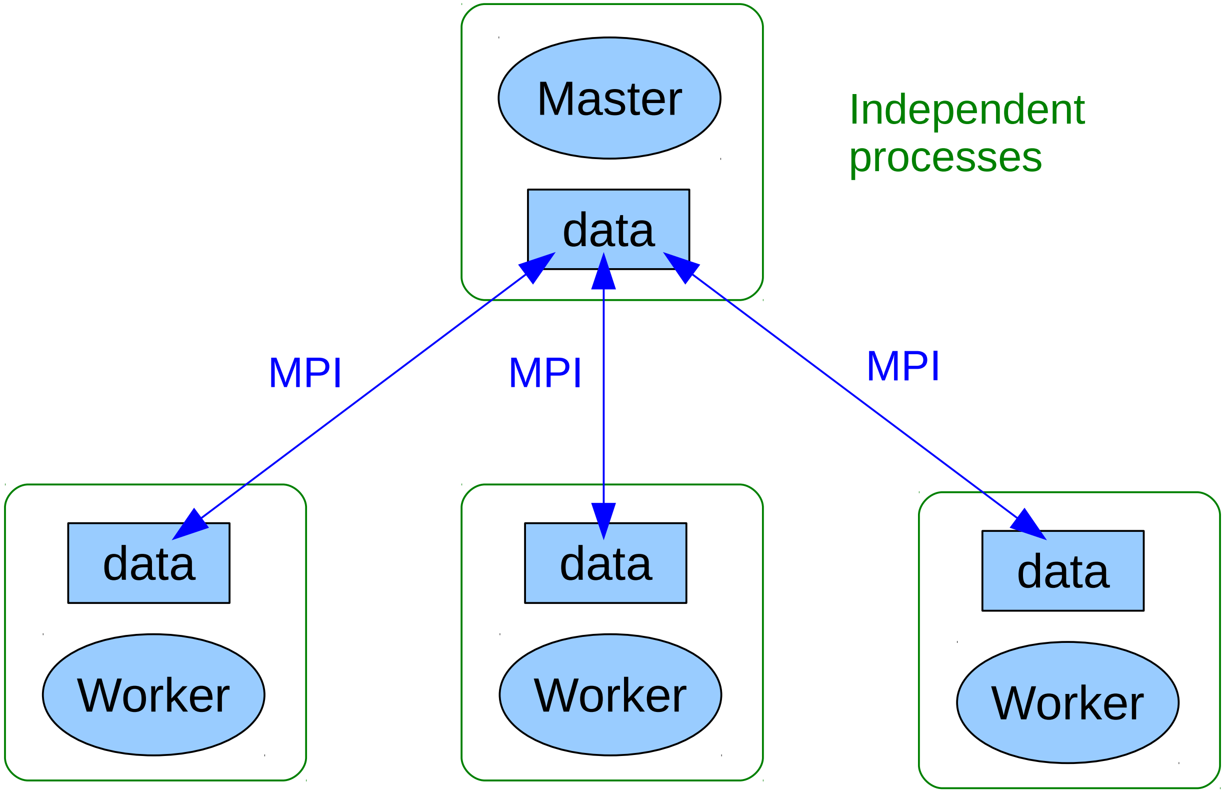
Karlsruhe, 1998-

Pre-SFB:

Fliegner, Retey, Vermaseren ’00

SFB:

Tentyukov, Fliegner, Frank, Onischenko, Retey, Staudenmaier, Vermaseren ’04;
Staudenmaier, Steinhauser, Tentyukov, Vermaseren ’06



Recent (Par)FORM Development

- ParFORM: implementation of missing features, bugfixes
- New features in the sequential FORM

  [FORM 4.0 Kuipers, TU, Vermaseren, Vollinga ’12
  FORM 4.1 Kuipers, TU, Vermaseren ’13]
- They should work also on TFORM and ParFORM as expected, at least must give correct results. (Otherwise: bugs.)

  Know issues : https://github.com/vermaseren/form/issues
- Parallelization is transparent. Just try

```
$ form myprogram.frm
```

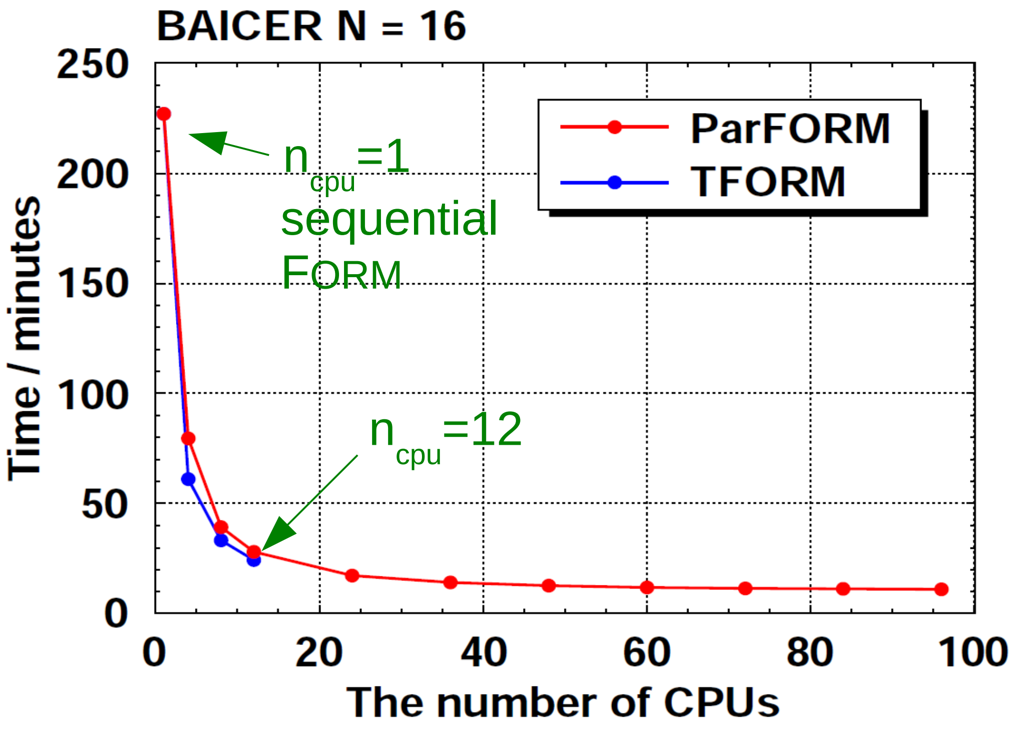
Need more speed on multi-core processors?

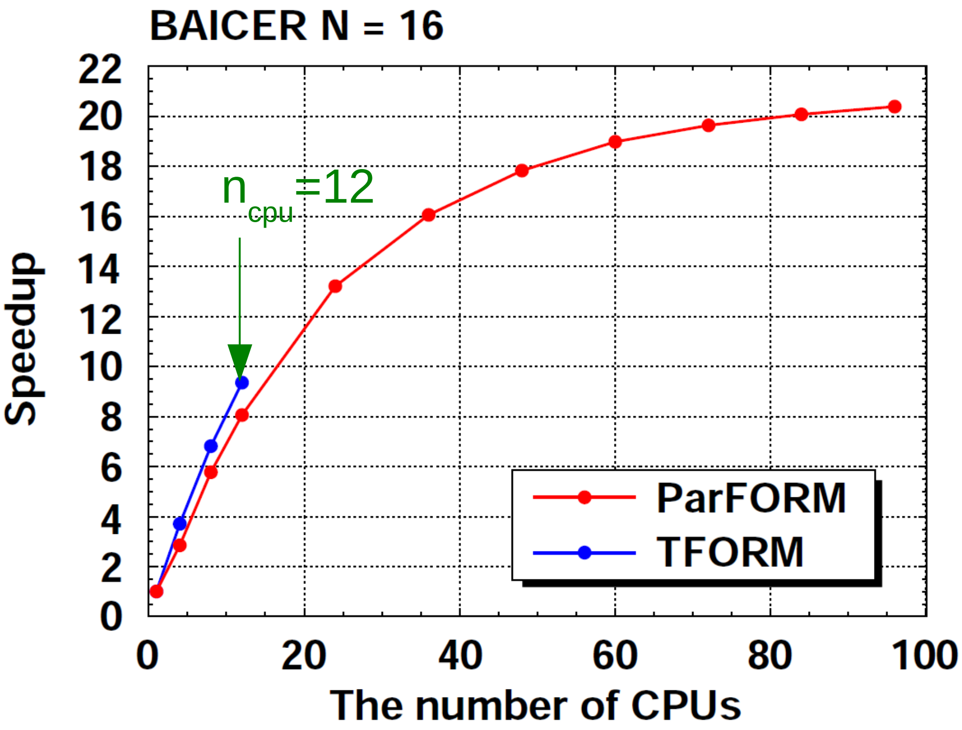
```
$ tform -w8 myprogram.frm
```

Need more speed and/or disk space on computer clusters?

```
$ mpirun -np 64 parform myprogram.frm
```

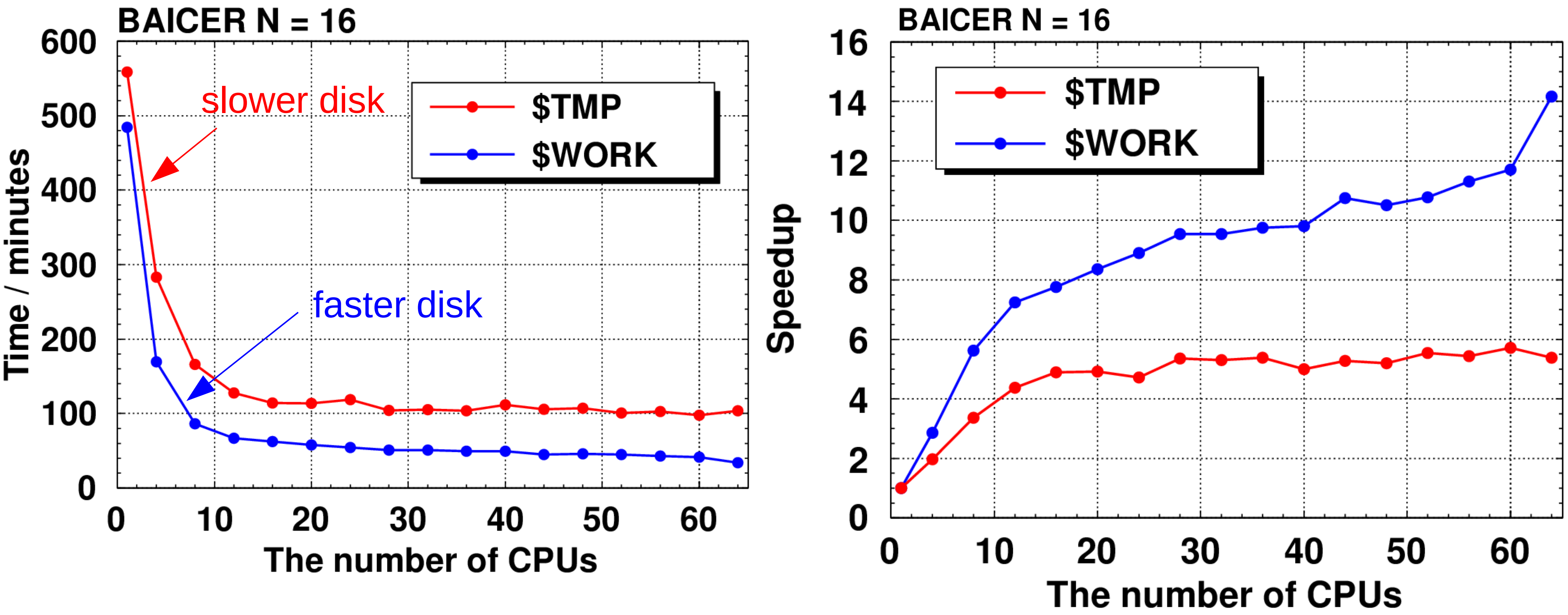
Benchmark (Comparison with TFORM)





- BAICER benchmark on ttpmoon cluster
  Each node has 12 cores (X5675 @ 3.07GHz),
  96 GB RAM, 3.6TB local disk (Raid 0 with 6 stripes)
  and connected by QDR Infiniband

Benchmark (Disk Speed Effect)



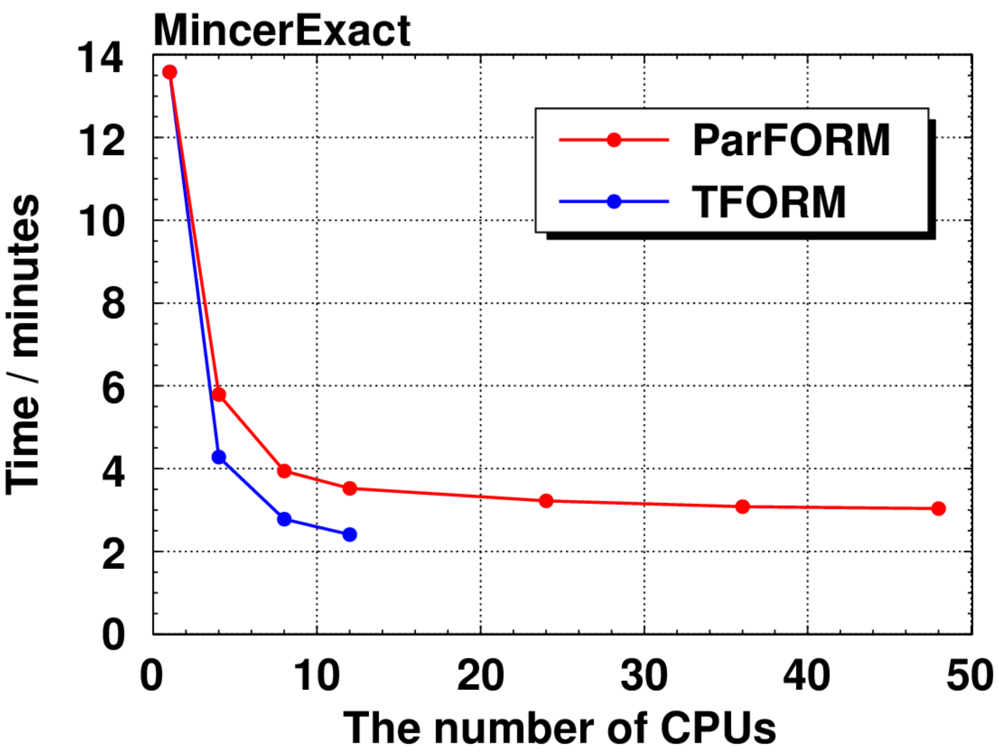
- BAICER benchmark with ParFORM on HP XC4000 at KIT SCC. Each node has 4 cores (AMD Opteron @ 2.6GHz)

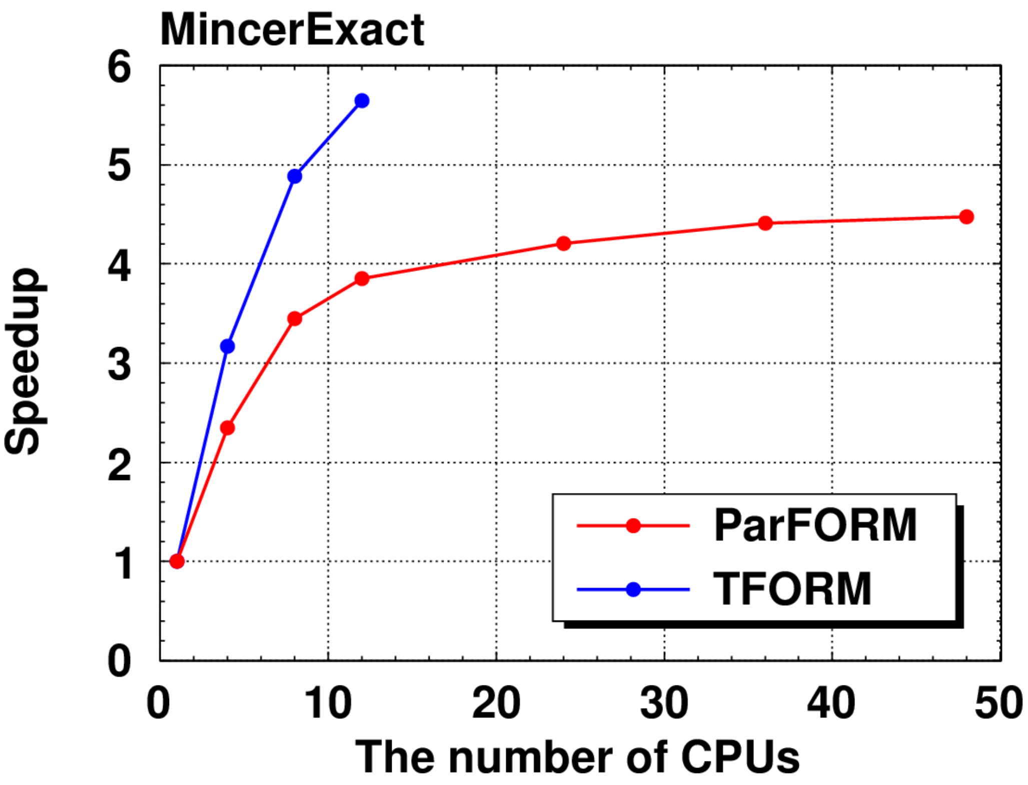
FORMTMP=$TMP: local disk (R/W perf. / node: 60/60MB/s)
$WORK: global disk (R/W perf. / node : 320/400MB/s)

Disk speed can considerably affect on the performance

Benchmark (MincerExact)

- A benchmark result of MincerExact, which heavily uses rational functions introduced in 4.0, on ttpmoon.
  Since the problem is not so big (14min by FORM), only small benefit. But ParFORM and TFORM work correctly.





FIRE

[Smirnov ’08; Smirnov^2 ’13 (FIRE 4); Smirnov ’14 (FIRE 5)]

- Feynman Integral Reduction

  Source code : https://bitbucket.org/feynmanIntegrals/fire

- Perform reduction of integrals to the master integrals using IBP relations by Laporta algorithm

  [Chetyrkin, Tkachov ’81] [Laporta ’00] Other software: AIR, REDUZE, Crusher (unpublished), etc.

- Latest version: FIRE 5
  - Front-end: Mathematica
  - Core part for the reduction written in C++
    - Performance improvement: 20+ times faster
  - Multithreading by the POSIX threads (Pthreads)
    - sectors of the same level, FERMAT processes [Lewis]
  - Can use LiteRed rules [Lee ’12]

FIESTA

[Smirnov, Tentyukov ’08; Smirnov^2, Tentyukov ’09 (FIESTA 2); Smirnov ’13 (FIESTA 3)]

- Feynman Integral Evaluation by a Sector decompositTion Approach [Binoth, Heinrich ’00; ’04, ...]

  Source code : https://bitbucket.org/feynmanIntegrals/fiesta

- Evaluate integrals (mainly) numerically

  Other software: sector-decomposition, SecDec, etc.

- Latest version: FIESTA 3
  - Algebraic part: Mathematica
  - Integrator written in C++; MPI parallelization
  - Physical region: contour deformation [Soper ’00, ...]
  - Asymptotic expansions (MB, Regions)
  - Implementation of a geometric strategy [Kaneko, TU ’10]

Summary

- ParFORM http://www.nikhef.nl/~form https://github.com/vermaseren/form
  - Parallel version of FORM based on MPI
  - Runs on, e.g., computer clusters
- FIRE http://science.sander.su/FIRE.htm https://bitbucket.org/feynmanIntegrals/fire
  - Performs IBP reductions
- FIESTA http://science.sander.su/FIESTA.htm https://bitbucket.org/feynmanIntegrals/fiesta
  - Evaluates Feynman integrals numerically

Backup Slides

Comparison with Other CAS

Mathematica, Maple, etc.

Swiss Army knife

- Much built-in mathematical knowledge (integration, solving equations, special functions etc.)
- Very general, versatile (sometimes overkill)
- Big and slow (especially on large problems)
- (Many of them are) proprietary

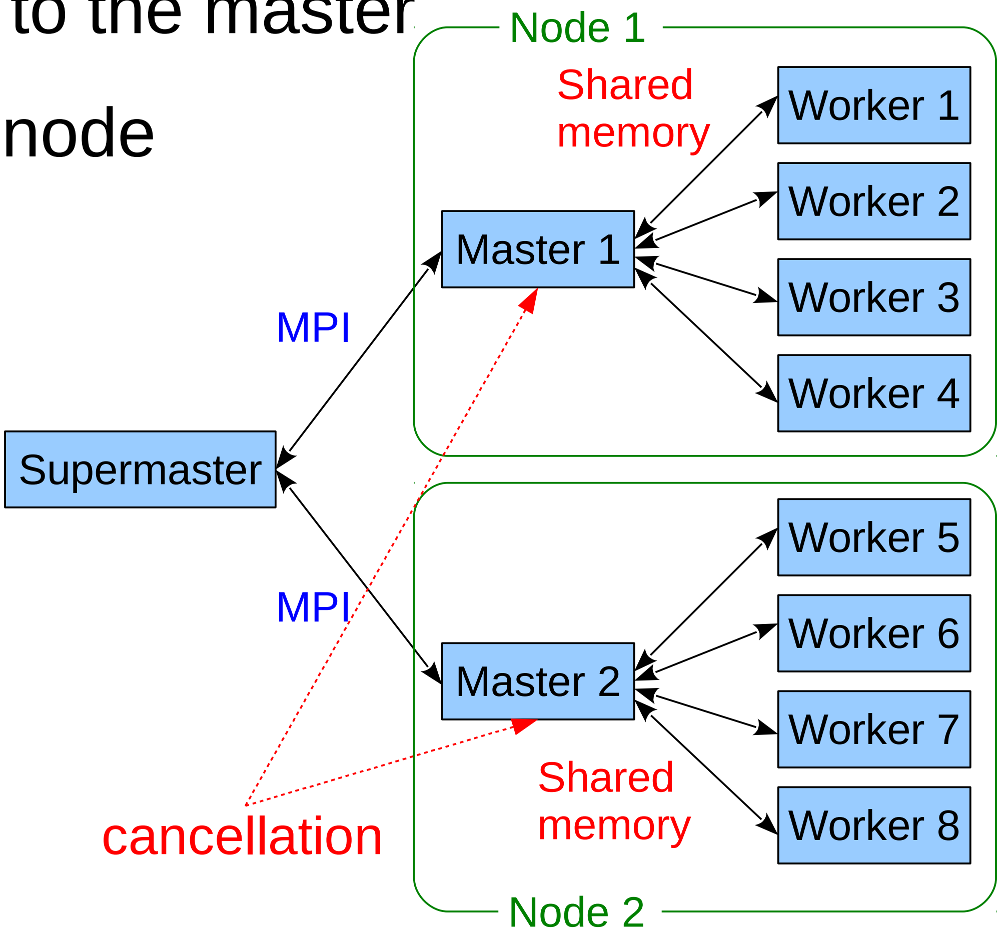
FORM

Chef's knife

- Limited built-in knowledge (calculus with tensors and gamma matrices, etc.)
- Optimized for efficiency
- Small and fast (also on large problems)
- Open source

Another way to FORM Parallelization?

- On computer clusters built from multicore processors:
  - Hybrid MPI/Pthreads parallelization
  - Avoid heavy network traffic to the master
  - No MPI overheads in each node


Node 1
Shared memory
Worker 1
Worker 2
Worker 3
Worker 4
Master 1
MPI
Supermaster
MPI
Master 2
Worker 5
Worker 6
Worker 7
Worker 8
Shared memory
cancellation
Node 2