UNIVERSITETET I OSLO

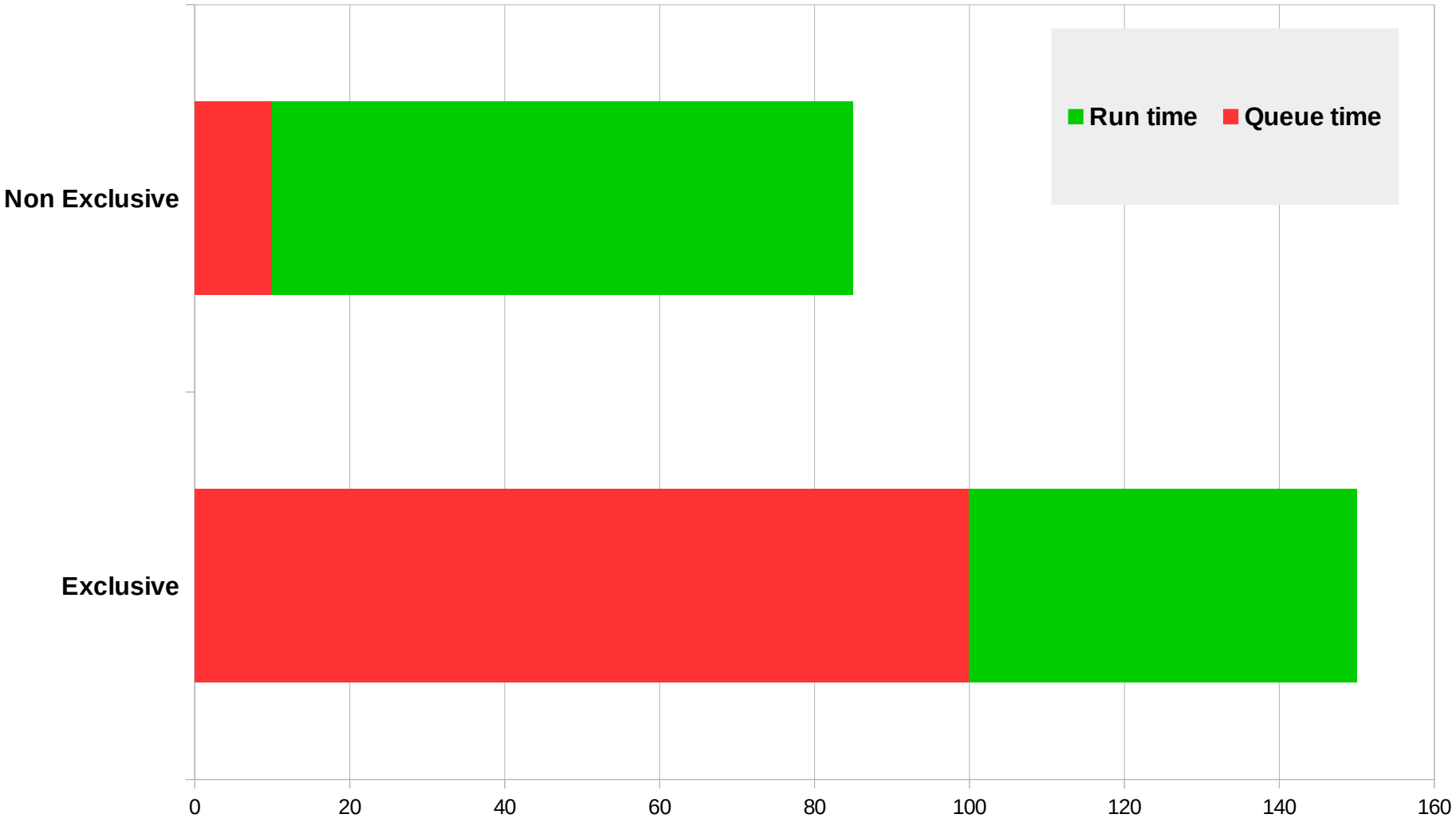
Cluster performance, how to get the most out of Abel

Ole W. Saastad, Dr.Scient
USIT / UAV / FI / FT
April 6th 2016

Introduction

- Smarte queueing
- Smart job setup
- Scaling
- Effective IO
- Big data handling
- Compiler usage
- Programming for the future

Smart queueing



Smart queueing

- Exclusive or none-exclusive ?
- Exclusive

  --ntasks-per-node=16 (or 20 for rack 31)
  --nodes=n
- None-exclusive

  --ntasks=m

Smart job setup

- Ask only for what you really need
- Stage data smart
- Carefully consider the IO load
- Have you control over scaling and input ?
- What happen if your job crashes or is killed ?
- What about log files ?

Ask only for what you need

- Queue system allocate and prioritise based on your footprint
- Footprint is based on
  - number of cores/nodes
  - Memory
  - Wall time

- Large footprint means longer wait time in queue

Stage data smart

- /work is faster and larger than /cluster
- There is NO backup on /work
- $SCRATCH is on /work
- $SCRATCH is deleted when job is finished
- /work/users/<your user name> is available
  - Files are kept for 45 days, no backup
- /project is slow, stage files on /work

Carefully consider the IO load

- /project is slow and not part of Abel
- /cluster is backed up and high performance
- /work is not backed up and has even higher performance
- Backup cannot tackle millions of files
- Small log files are ok on /cluster/home
- Large files should be on /work
  - $SCRATCH for temporary scratch files
  - /work/user/<username> for log files
- Files on /work/users are **deleted** after 45 days
  - Touching the files does not prevent this !

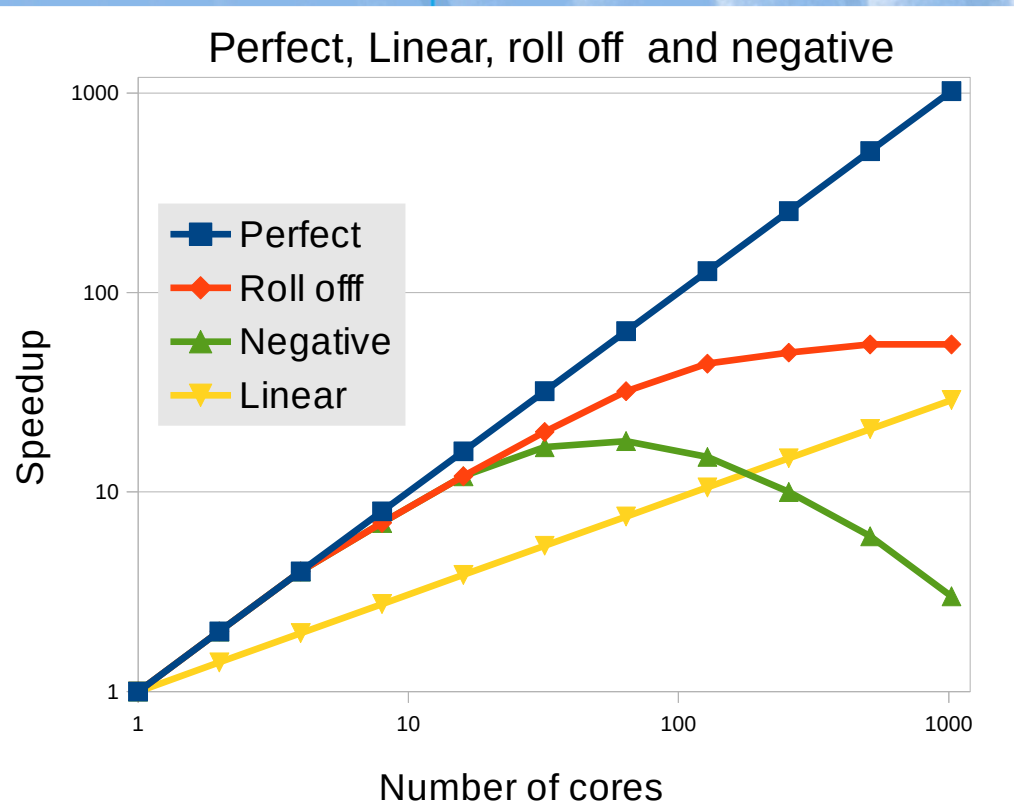
Carefully consider the IO load

- /cluster and /work file system are unhappy with millions of small files
- If you use millions of temporary files ask for local scratch named $LOCALTMP
  - --gres=localtmp:x (x is in GiB)
- Backup system are unhappy with millions of small files
  - Use a directory called nobackup or /work

Have you control over scaling and input ?

- Do your application scale ?
- Does it scale to the core count you request?
- Does it scale with the given input ?

- OpenMP threads
- MPI ranks
- Hybrid (multi threaded MPI executables)



What happen if your job crashes or is killed ?

- If you application crashes everything works as if it terminated normally

- If your job is killed (memory, time limit etc)
  - chkfile <file> work as expected
  - Directory $SCRATCH is removed
  - Files on $SCRATCH are lost

What about log files ?

- Log files can reside on /cluster/home or /work
  - No need to use $SCRATCH for them
  - Logs of reasonable size on $SUBMITDIR
- If log files are written to $SCRATCH remember to use chkfile
- Log files are required when reporting problems (RT)

Effective IO

- How much IO is your application doing ?
- Few large files or millions of small files ?
- Where are the files placed ?
- Copying of files
- Archiving many files to one large
- Do the files compress ?
  - Not all files compress well

Effective IO

- /work and /cluster are global parallel file systems, BeeGFS.
- In order to make it coherent locks and tokens are needed.
- Any operation must be locked to prevent corruption if more than one writer want to update a file
- All parallel file systems struggle to cope with huge amount of metadata updates.
- They love few large files !

Effective IO

- Use $SCRATCH or /work/users/ during runs
- Use $LOCALTMP for millions of files
  - Make a tar archive of these before moving them to /work or /cluster
- Stage files from /projects to /work/users
  - /project has limited bandwidth

Big data handling

- Few large files or millions of small files ?
- BeeGFS can handle large files easy
  - Bandwidth like 8-10 GiB/s
  - /work have higher bandwidth than /cluster
- Use infiniband if possible
  - scp file.ext <host ib ip number>:
  - Doubles the transfer speed

Big data handling

- Many small files are problematic
- Backup system struggle with millions of files
- BeeGFS struggle with metadata of millions of entries

- Make archives of directories with large number of files
- Use local scratch or temporal storage of possible
  - These sizes are relatively small
  - Tar directory to a single large file

- Mount an image on /work as a file system
  - Require help from support team

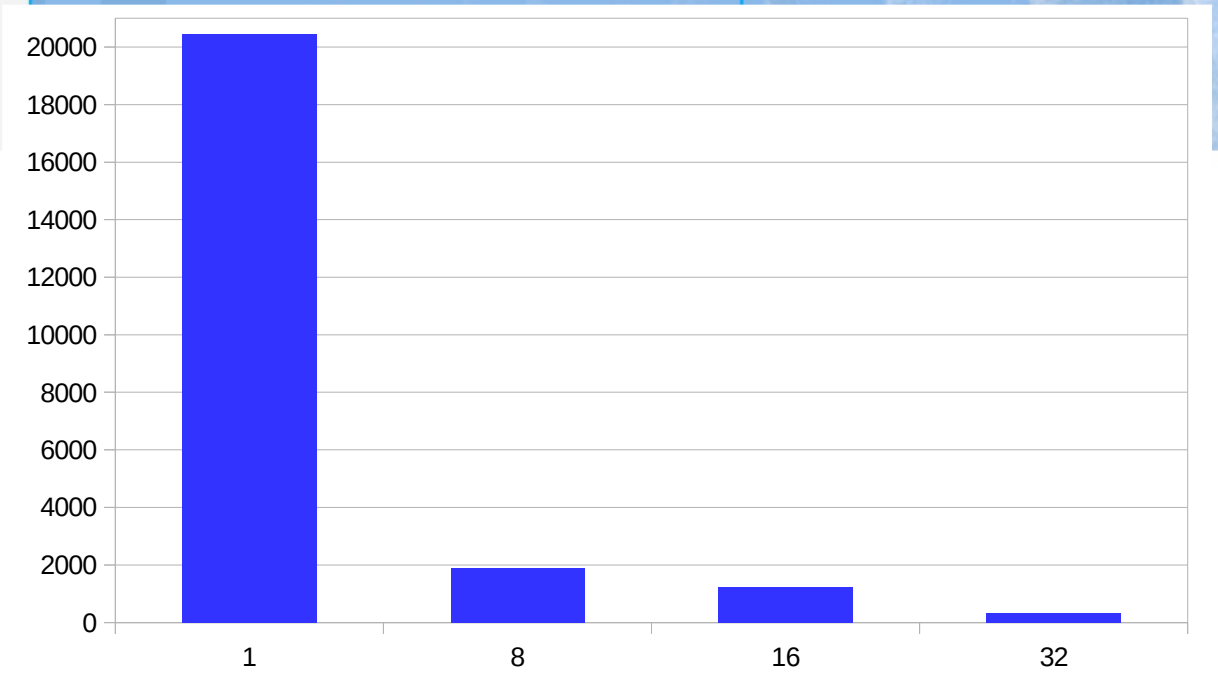
UNIVERSITETET I OSLO

Big data handling

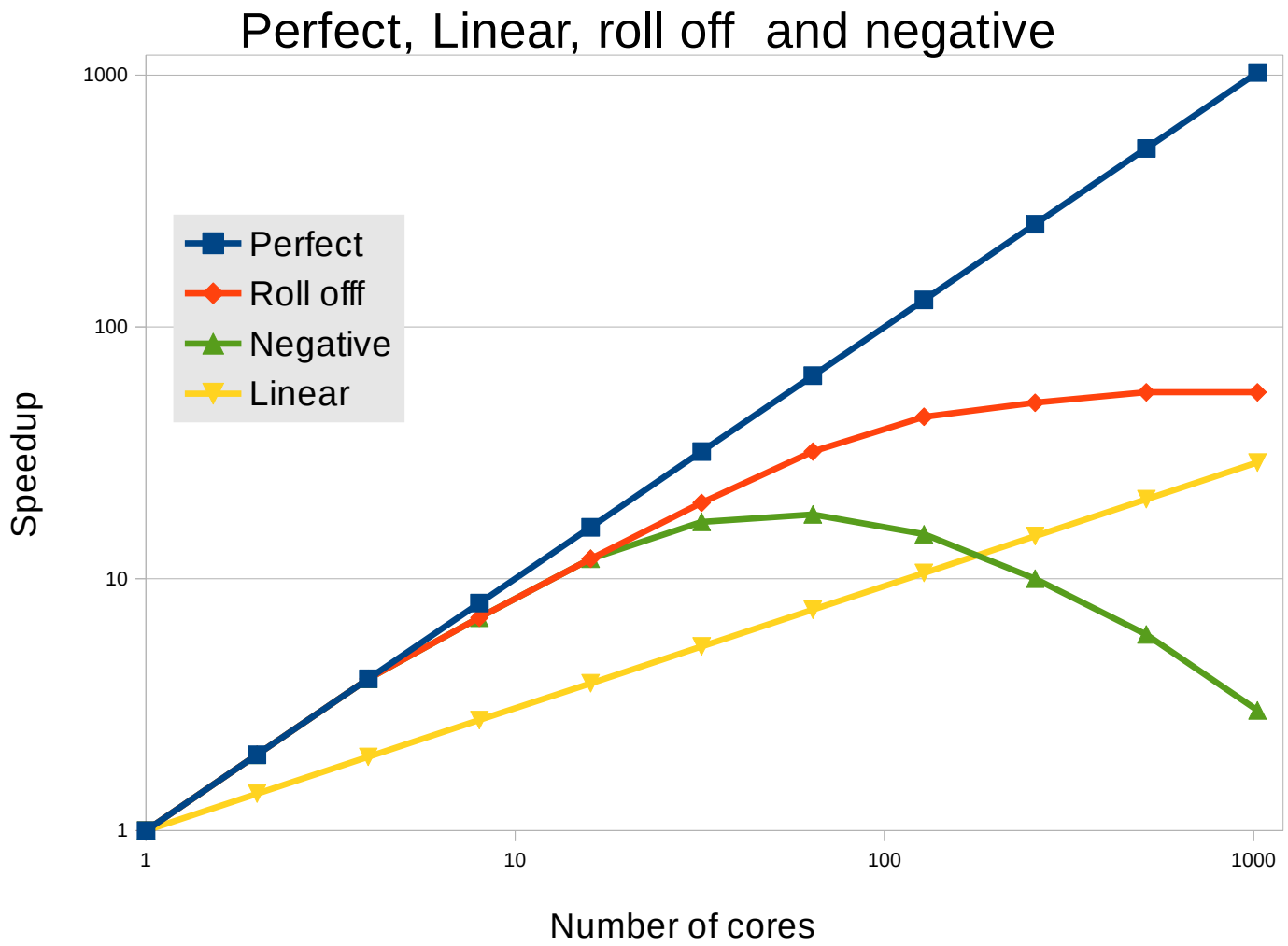
- Copying of millions of files are slow
- Rsync may take hours
- Option to use parallel rsync

1: rsync

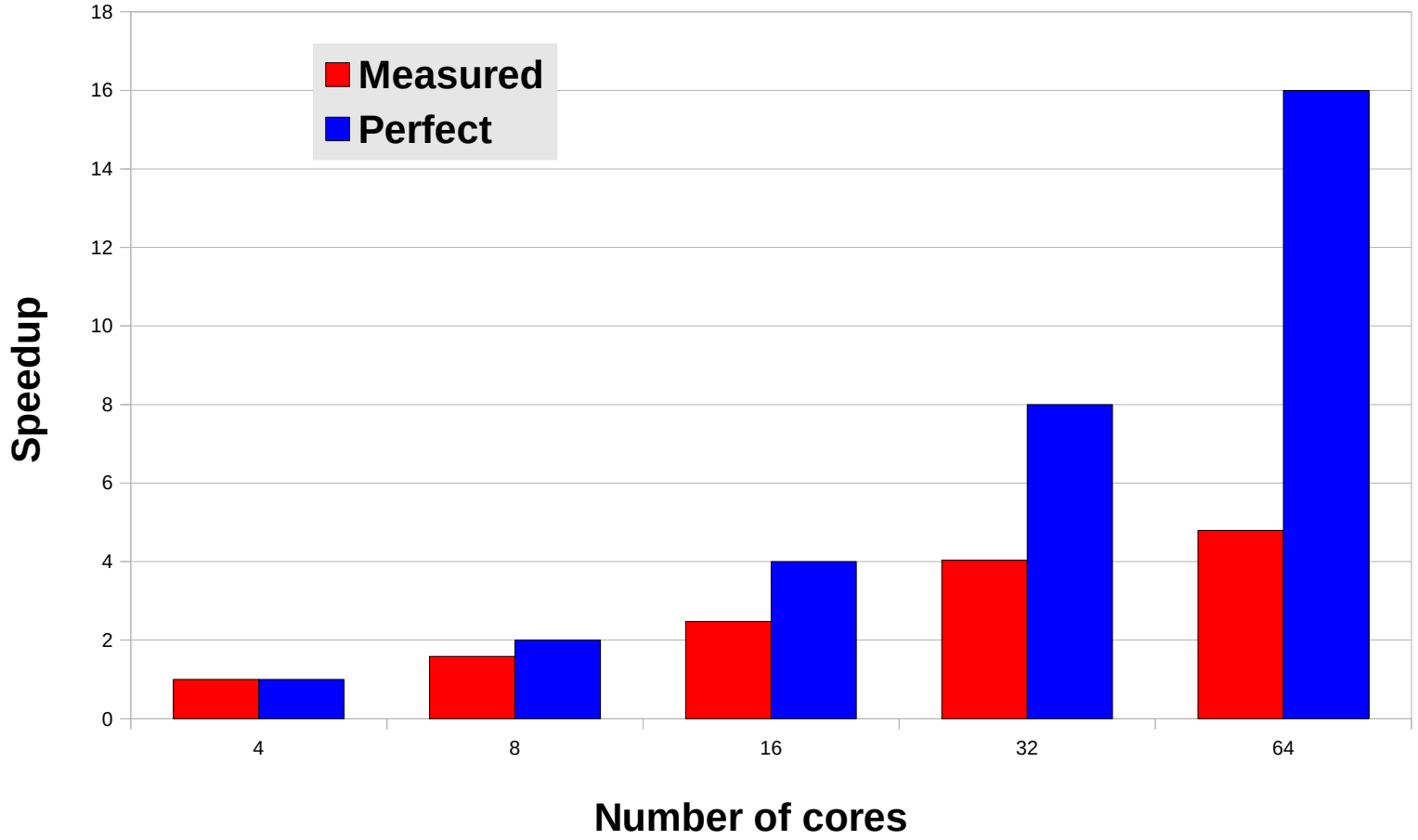
2-32 : parsync



Scaling


Perfect, Linear, roll off and negative
1000
100
10
1
Speedup
Perfect
Roll offf
Negative
Linear
1
10
100
1000
Number of cores

Scaling

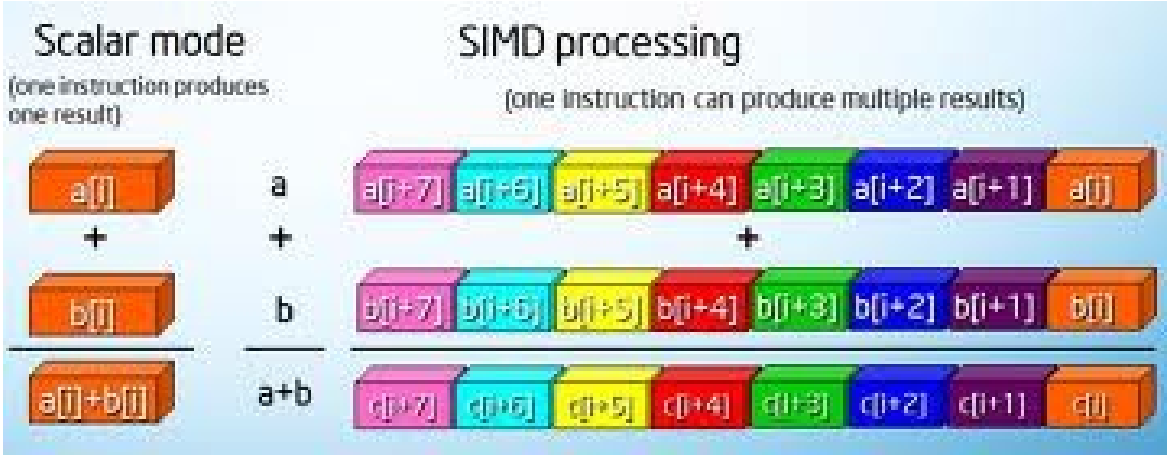

Measured
Perfect
Speedup
0
2
4
6
8
10
12
14
16
18
4
8
16
32
64
Number of cores

Scaling

- How does your application scale ?
- Is it dependent on input ?
- Are you running at the sweet spot ?
- Threads ?
- MPI ?
- Hybrid ?
- Will your application scale to more than 120 threads to use the new Intel Knights Landing processor ?

Compiler usage

- Parallel – 16 to 20 cores in a node
  - Vectorisation
  - OpenMP
  - MPI
- More and more focus on vectorisation

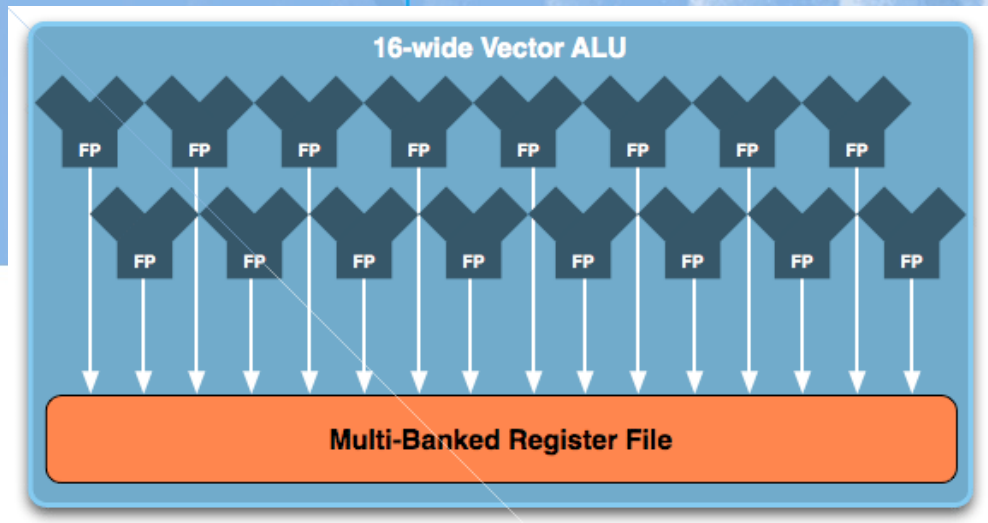


Programming for the future

- Intel Many Integrated Cores, MIC
  - Knight Corner (KC) installed
  - Knights Landing (KL) later 2016
  - KC : 1 Tflops/s
  - KL : 3 Tflops/s
  - Same architecture x86-64 (Silvermont core)
  - Same compilers
  - Same libraries

Programming for the future

- Intel Knight Landing 72 cores and 288 theads
  - Need at least 144 threads → OpenMP+MPI
  - 512 bits vector unit
  - Fused Multiply add



Fused Multiply-Add (FMA)

A × B = Product (retain all digits)

\+

C = Result

Programming for the future

- Pure MPI may yield too many ranks
  - Large memory demand
- Pure threads may scale weakly
  - OpenMP scaleing is often limited
- Hybrid multi threaded MPI

- One MPI rank per core, 2-4 threads per core

Programming for the future

- Vectorization vital for performance
  - Scalar code perform at 1/16 performance
- OpenMP 4.0 introduces SIMD directives
- All OpenMP programming facilities now apply to SIMD
- Vectorization is made easier with OpenMP, just like threading was made simpler.

Programming for the future

- OpenMP SIMD - vectorization

- Single program, multiple data
- Vector units are becoming increasingly more important
- AVX / AVX2 256 bits = 4 dp and 8 sp floats
- MIC has 512 bits vector = 8 dp and 16 sp floats
- Vector fused multiply add double vector performance

Programming – OpenMP 4.0

- OpenMP 4.0 SIMD

```
!$omp simd reduction(+:sum)
 do i = 1,n
    x = h*(i-0.5)
    sum = sum + f(x)
 enddo
 pi = h*sum
```

Programming - OpenMP 4.0

- OpenMP 4.0 SIMD

```
!$omp simd reduction(+:sum)
 do i = 1,n
    x = h*(i-0.5)
    sum = sum + f(x)
 enddo
 pi = h*sum
```

Programming – OpenMP 4.0

- OpenMP 4.0 SIMD

```
 module func_f
  contains
!$omp declare simd f
    function f(x)
      real(8) :: f
      real(8) :: x
      f = (4/(1+x*x))
    end function f
end module func_f
```

Programming - Intel tuning tools

- Tuning tools

- Compiler – *only time the source code is read*
- MPI library – *can provide vital statistics*
- Vector Advisor - *vectorization*
- Thread Inspector - *Threads*
- Vtune Amplifier – *CPU / memory performance*
- Trace Analyser – *MPI calls*

Programming - Intel tuning tools

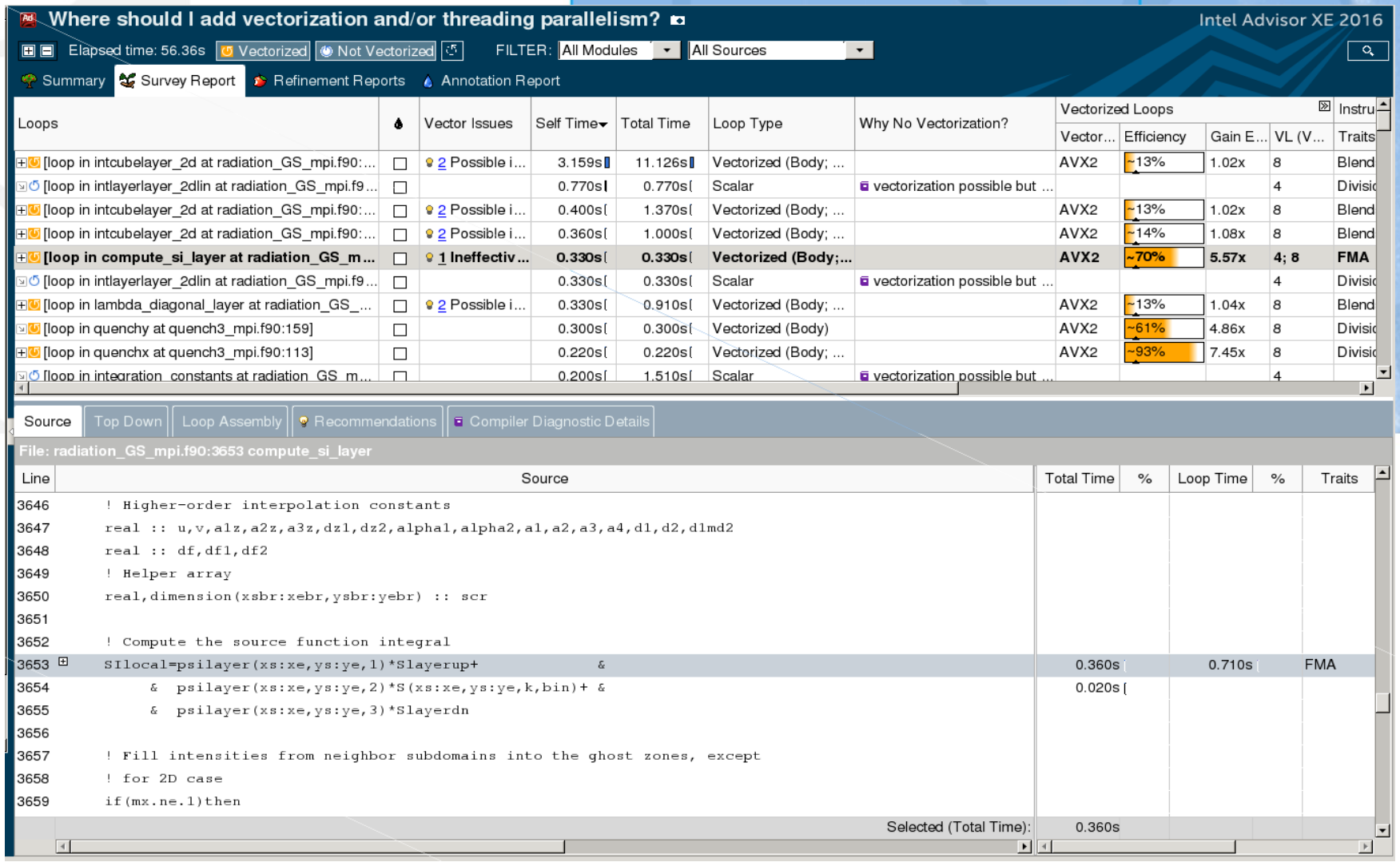
- Compiler – *only time the source code is read*
- -qopt-report5 -qopt-report-phase=vec
- Read the *.optrpt files !

Programming - Intel tuning tools

- Vector Advisor
- Checks the program for vectorization
- Checks performance scalar vs. vector
- Checks loop performance

UNIVERSITETET I OSLO

Intel XE-Advisor

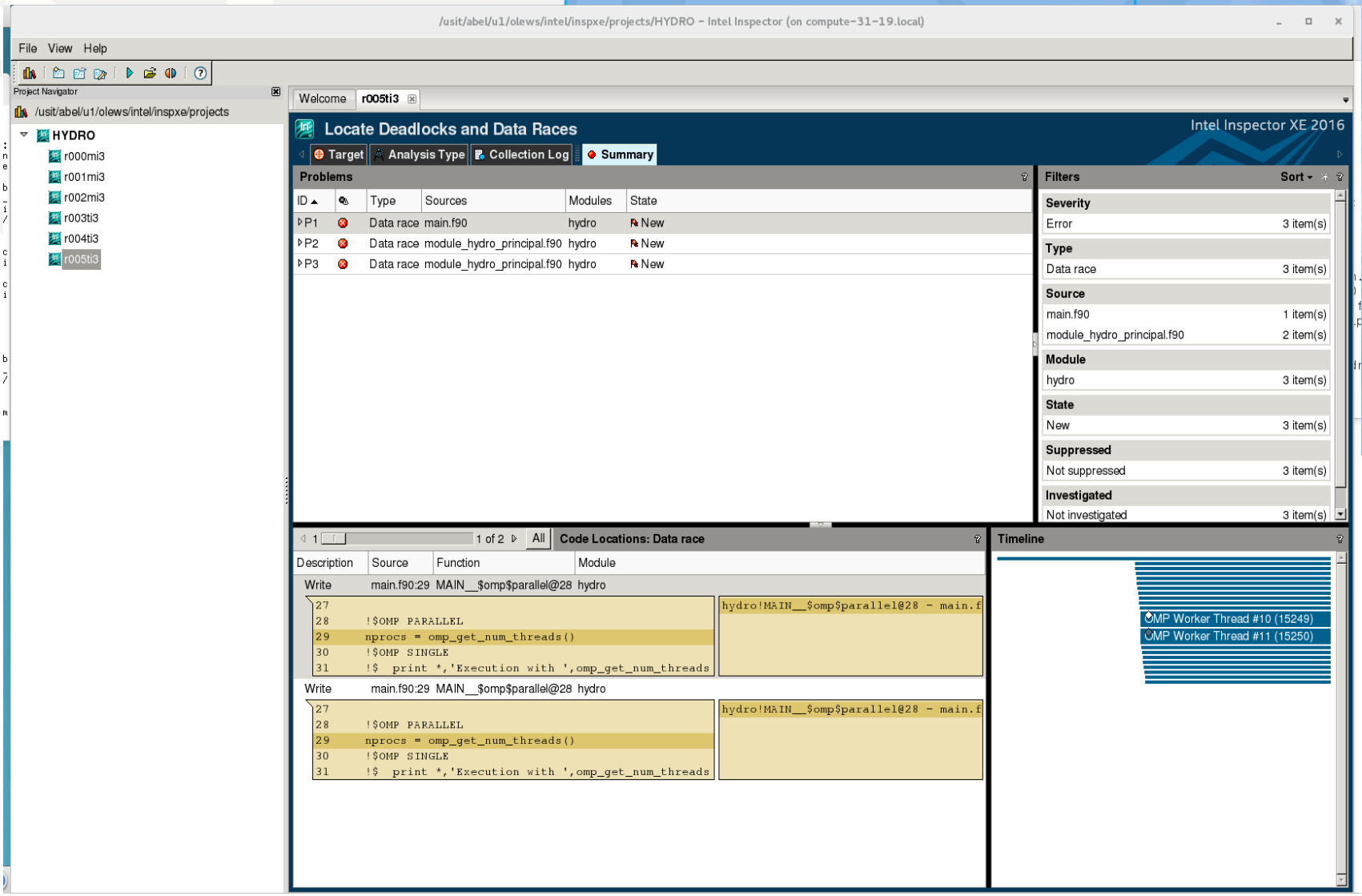


Programming - Intel tuning tools

- XE thread Inspector
- Checks the theading in the program
- Checks the loops and threads

UNIVERSITETET I OSLO

Intel XE-Inspector

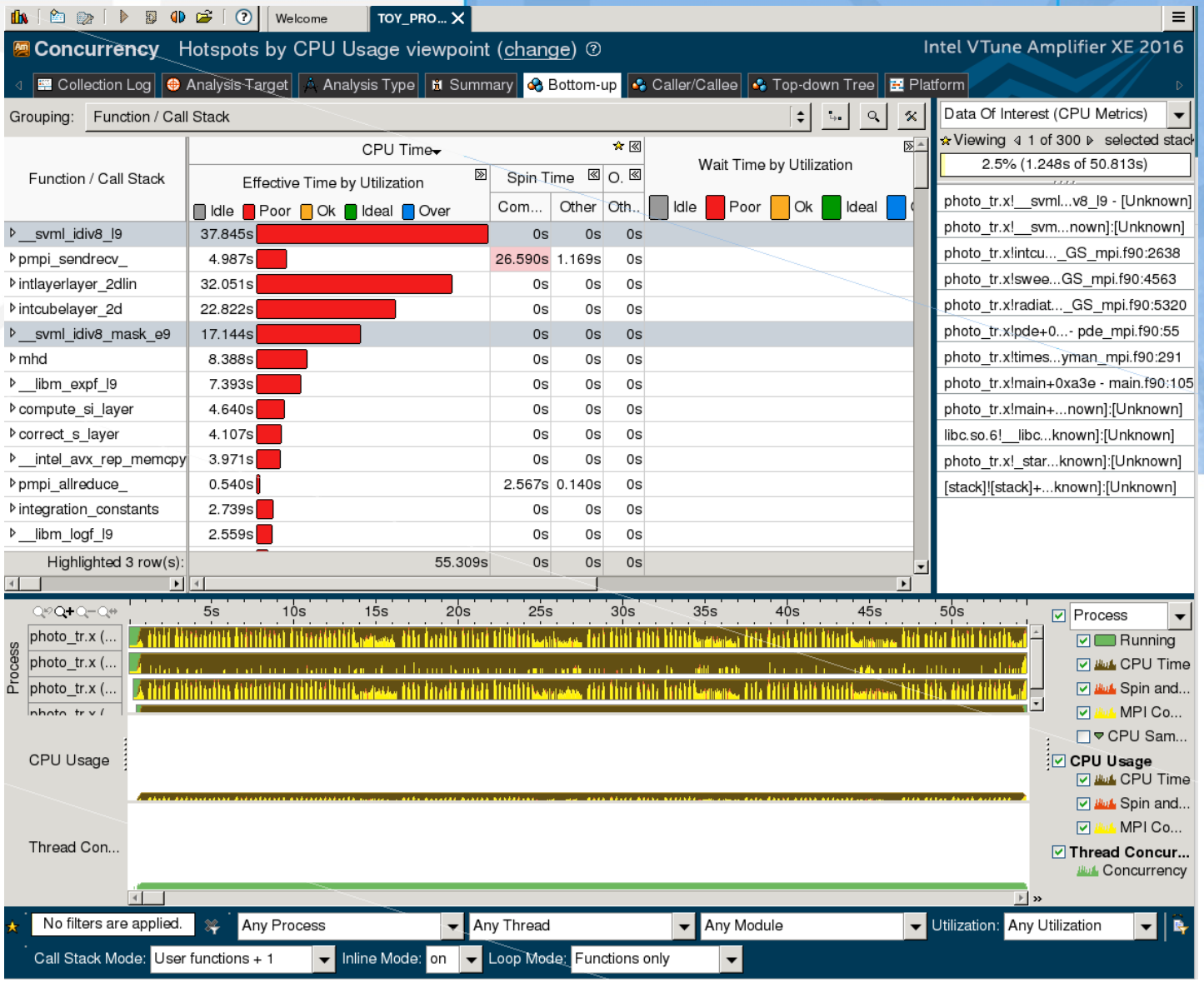


Programming - Intel tuning tools

- XE Vtune Amplifier
- Checks the CPU performance
- Originally a tool for CPU design
- Provide a large range of metrics
- Instructions, cache, memory usage

UNIVERSITETET I OSLO

Intel Vtune-Amplifier

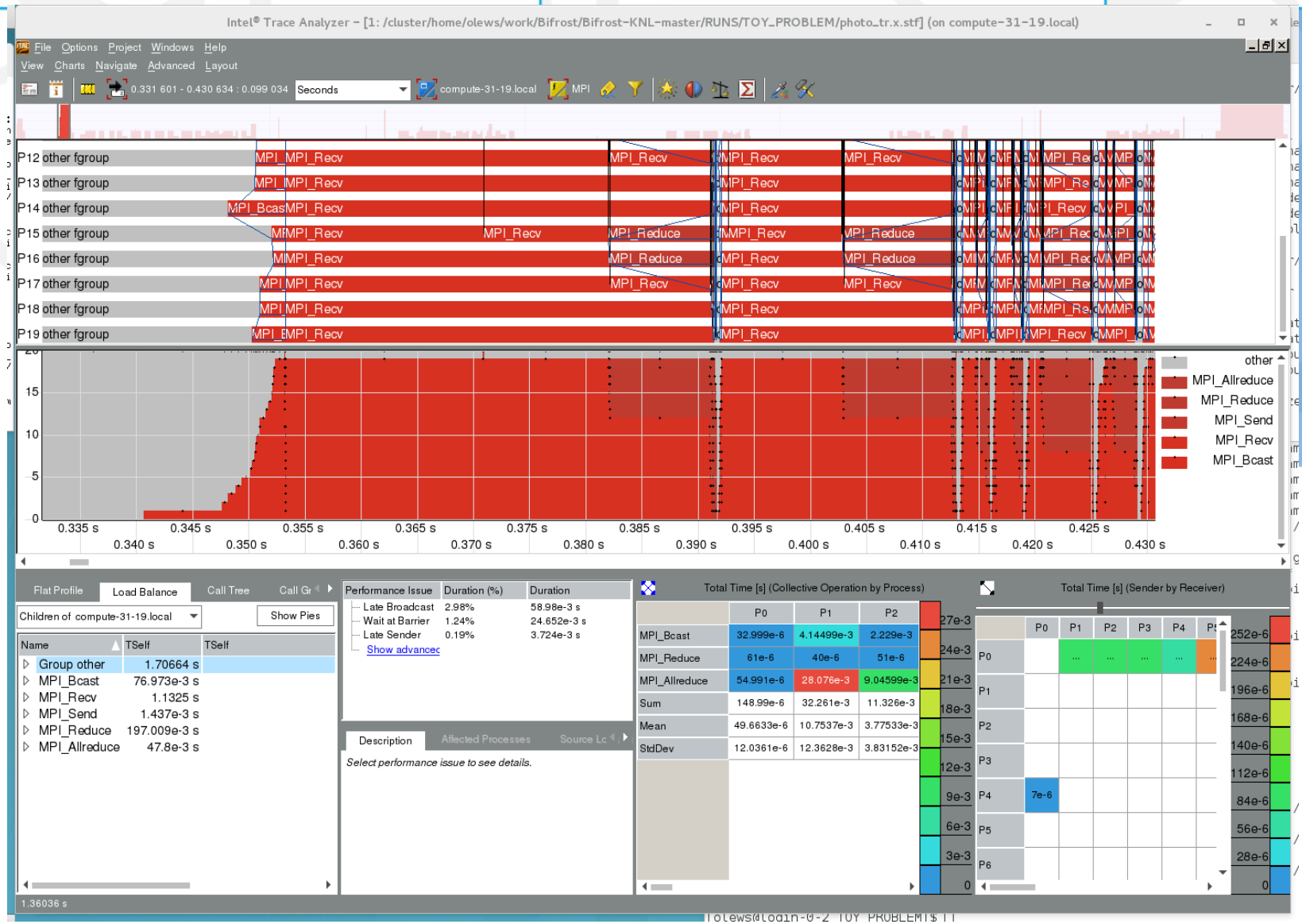


Programming - Intel tuning tools

- Trace Analyser
- Checks the MPI communication
- Trace all MPI calls

UNIVERSITETET I OSLO

Intel Trace-Analyser



Intel tuning tools