Some Results Obtained

Hendryk Bockelmann, DKRZ

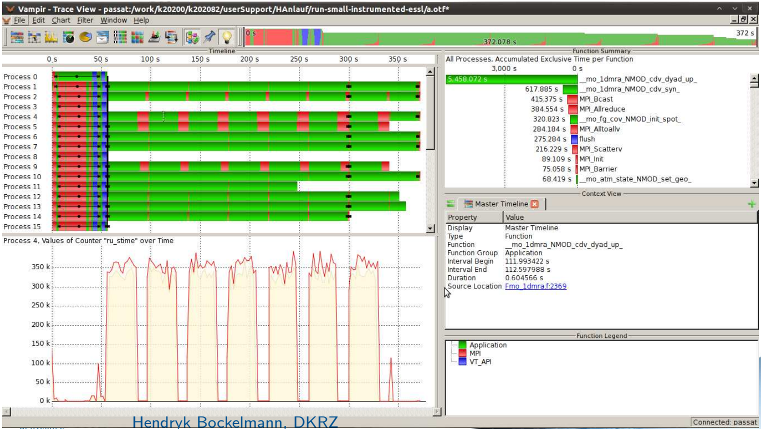
Problem

"while testing some code that ran perfectly on our old POWER5, I noticed a very bad performance on blizzard. rusage/hpccount gave the following output:"

```
Metrik              [Unit]   :     Average    Minimum    Maximum
maximum RSS         [Kbytes]:   1121529.67  1059896.0  2015480.0
time in user mode   [sec]    :      804.57     662.59     903.04
time in system mode [sec]    :     2703.09    2608.43    2860.22
inst per run cycle           :        0.42       0.42       0.43
peak performance           % :        0.02       0.01       0.03
```

Locate high sys time

- instrument code with vampirtrace and use VT_RUSAGE=all
- high sys time within module mo_1dmra localized in subroutine cdv_dyad_up



All Processes, Accumulated Exclusive Time per Function

Time	Function
5,458.072 s	__mo_1dmra_NMOD_cdv_dyad_up_
617.885 s	__mo_1dmra_NMOD_cdv_syn_
415.375 s	MPI_Bcast
384.554 s	MPI_Allreduce
320.823 s	__mo_fg_cov_NMOD_init_spot_
284.184 s	MPI_Alltoallv
275.284 s	flush
216.229 s	MPI_Scatterv
89.109 s	MPI_Init
75.058 s	MPI_Barrier
68.419 s	__mo_atm_state_NMOD_set_geo_

Property	Value
Display	Master Timeline
Type	Function
Function	__mo_1dmra_NMOD_cdv_dyad_up_
Function Group	Application
Interval Begin	111.993422 s
Interval End	112.597988 s
Duration	0.604566 s
Source Location	Fmo_1dmra.f:2369

- 🟩 Application
- 🟥 MPI
- 🟦 VT_API

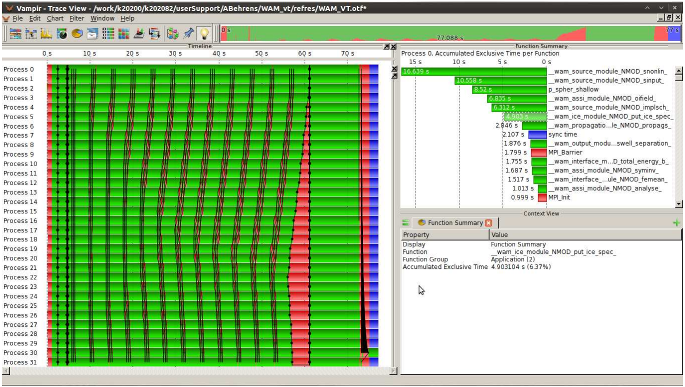
Solution

- MATMUL uses multithreading by default:

  > *"The default value for* num_threads *when using the* MATMUL *intrinsic equals the number of processors online. Changing the number of threads available to the* MATMUL *and* RANDOM_NUMBER *intrinsic procedures can influence performance."*

- set XLFRTEOPTS="intrinthds=1"

MPI barrier slows down ...



Vampir - Trace View - /work/k20200/k202082/userSupport/ABehrens/WAM_vt/refres/WAM_VT.otf*

Process 0, Accumulated Exclusive Time per Function

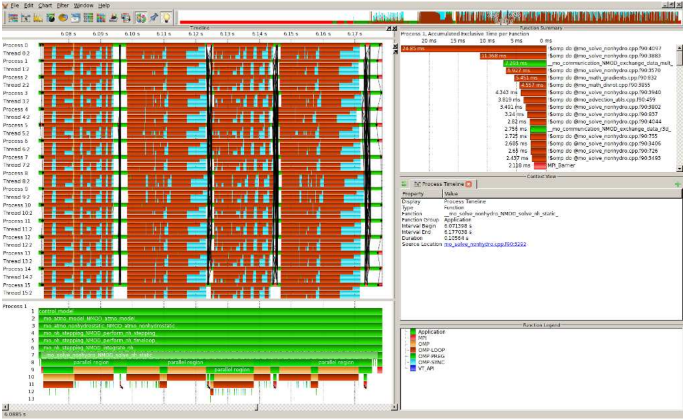
16.639 s	__wam_source_module_NMOD_snonlin_
10.558 s	__wam_source_module_NMOD_sinput_
8.52 s	p_spher_shallow
6.835 s	__wam_assi_module_NMOD_oifield_
6.312 s	__wam_source_module_NMOD_implsch_
4.903 s	__wam_ice_module_NMOD_put_ice_spec_
2.846 s	__wam_propagatio...le_NMOD_propags_
2.107 s	sync time
1.876 s	__wam_output_modu...swell_separation_
1.799 s	MPI_Barrier
1.755 s	__wam_interface_m...D_total_energy_b_
1.687 s	__wam_assi_module_NMOD_syminv_
1.517 s	__wam_interface_...ule_NMOD_femean_
1.013 s	__wam_assi_module_NMOD_analyse_
0.999 s	MPI_Init

Property	Value
Display	Function Summary
Function	__wam_ice_module_NMOD_put_ice_spec_
Function Group	Application (2)
Accumulated Exclusive Time	4.903104 s (6.37%)

solution: cpu-load due to additional ICE computations propagate with each nearest neighbour exchange

OpenMP schedule optimized

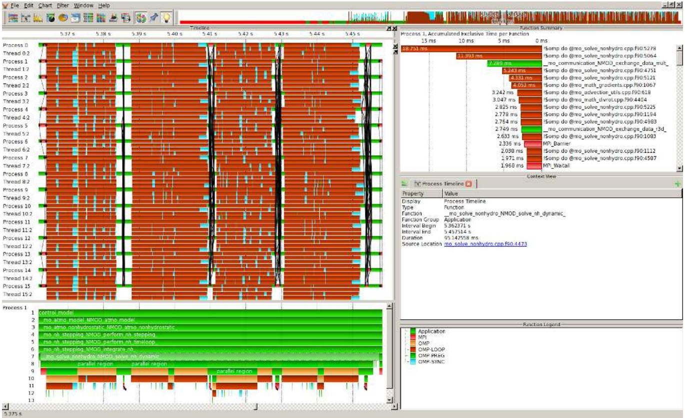
what is the best setup of OpenMP threads in a hybrid code?


File Edit Chart Filter Window Help
Timeline
6.08 s
6.09 s
6.10 s
6.11 s
6.12 s
6.13 s
6.14 s
6.15 s
6.16 s
6.17 s
Process 0
Thread 0:2
Process 1
Thread 1:2
Process 2
Thread 2:2
Process 3
Thread 3:2
Process 4
Thread 4:2
Process 5
Thread 5:2
Process 6
Thread 6:2
Process 7
Thread 7:2
Process 8
Thread 8:2
Process 9
Thread 9:2
Process 10
Thread 10:2
Process 11
Thread 11:2
Process 12
Thread 12:2
Process 13
Thread 13:2
Process 14
Thread 14:2
Process 15
Thread 15:2
Process 1
1 control_model
2 _mo_atmo_model_NMOD_atmo_model
3 _mo_atmo_nonhydrostatic_NMOD_atmo_nonhydrostatic_
4 _mo_nh_stepping_NMOD_perform_nh_stepping_
5 _mo_nh_stepping_NMOD_perform_nh_timeloop_
6 _mo_nh_stepping_NMOD_integrate_nh_
7 _mo_solve_nonhydro_NMOD_solve_nh_static_
8 parallel region parallel region parallel region
9 parallel region
10
11
12
13
6.0885 s
Function Summary
Process 1, Accumulated Exclusive Time per Function
20 ms
15 ms
10 ms
5 ms
0 ms
24.85 ms $omp do @mo_solve_nonhydro.cpp.f90:4097
11.368 ms $omp do @mo_solve_nonhydro.cpp.f90:3883
7.293 ms __mo_communication_NMOD_exchange_data_mult_
6.927 ms $omp do @mo_solve_nonhydro.cpp.f90:3570
5.451 ms $omp do @mo_math_gradients.cpp.f90:832
4.557 ms $omp do @mo_math_divrot.cpp.f90:3855
4.343 ms $omp do @mo_solve_nonhydro.cpp.f90:3940
3.819 ms $omp do @mo_advection_utils.cpp.f90:459
3.491 ms $omp do @mo_solve_nonhydro.cpp.f90:3802
3.24 ms $omp do @mo_solve_nonhydro.cpp.f90:837
2.82 ms $omp do @mo_solve_nonhydro.cpp.f90:4044
2.756 ms __mo_communication_NMOD_exchange_data_r3d_
2.725 ms $omp do @mo_solve_nonhydro.cpp.f90:755
2.605 ms $omp do @mo_solve_nonhydro.cpp.f90:3406
2.65 ms $omp do @mo_solve_nonhydro.cpp.f90:726
2.437 ms $omp do @mo_solve_nonhydro.cpp.f90:3493
2.118 ms MPI_Barrier
Context View
Process Timeline
Function Legend
Application
MPI
OMP
OMP-LOOP
OMP-PREG
OMP-SYNC
VT_API

Property	Value
Display	Process Timeline
Type	Function
Function	__mo_solve_nonhydro_NMOD_solve_nh_static_
Function Group	Application
Interval Begin	6.071398 s
Interval End	6.177038 s
Duration	0.10564 s
Source Location	mo_solve_nonhydro.cpp.f90:3292

nproma=16, nchunksize=2, static schedule (default)

OpenMP schedule optimized



Process 1, Accumulated Exclusive Time per Function

15 ms 10 ms 5 ms 0 ms	
18.751 ms	!$omp do @mo_solve_nonhydro.cpp.f90:5278
11.393 ms	!$omp do @mo_solve_nonhydro.cpp.f90:5064
7.289 ms	__mo_communication_NMOD_exchange_data_mult_
5.243 ms	!$omp do @mo_solve_nonhydro.cpp.f90:4751
4.331 ms	!$omp do @mo_solve_nonhydro.cpp.f90:5121
4.052 ms	!$omp do @mo_math_gradients.cpp.f90:1067
3.242 ms	!$omp do @mo_advection_utils.cpp.f90:618
3.047 ms	!$omp do @mo_math_divrot.cpp.f90:4404
2.825 ms	!$omp do @mo_solve_nonhydro.cpp.f90:5225
2.778 ms	!$omp do @mo_solve_nonhydro.cpp.f90:1194
2.764 ms	!$omp do @mo_solve_nonhydro.cpp.f90:4983
2.749 ms	__mo_communication_NMOD_exchange_data_r3d_
2.633 ms	!$omp do @mo_solve_nonhydro.cpp.f90:1083
2.336 ms	MPI_Barrier
2.038 ms	!$omp do @mo_solve_nonhydro.cpp.f90:1112
1.971 ms	!$omp do @mo_solve_nonhydro.cpp.f90:4587
1.968 ms	MPI_Waitall

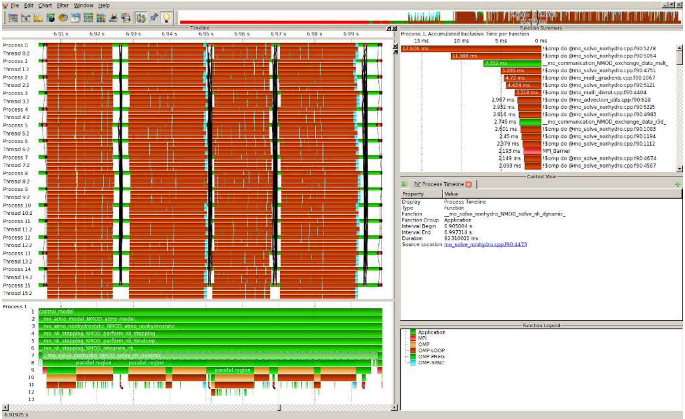
Property	Value
Display	Process Timeline
Type	Function
Function	__mo_solve_nonhydro_NMOD_solve_nh_dynamic_
Function Group	Application
Interval Begin	5.362371 s
Interval End	5.457514 s
Duration	95.142558 ms
Source Location	mo_solve_nonhydro.cpp.f90:4473

- 🟩 Application
- 🟥 MPI
- 🟧 OMP
- 🟥 OMP-LOOP
- 🟩 OMP-PREG
- 🟦 OMP-SYNC

nproma=16, nchunksize=1, dynamic schedule

Hendryk Bockelmann, DKRZ

OpenMP schedule optimized



Process 1, Accumulated Exclusive Time per Function

17.606 ms	!$omp do @mo_solve_nonhydro.cpp.f90:5278
11.388 ms	!$omp do @mo_solve_nonhydro.cpp.f90:5064
7.252 ms	__mo_communication_NMOD_exchange_data_mult_
5.095 ms	!$omp do @mo_solve_nonhydro.cpp.f90:4751
4.72 ms	!$omp do @mo_math_gradients.cpp.f90:1067
4.434 ms	!$omp do @mo_solve_nonhydro.cpp.f90:5121
3.318 ms	!$omp do @mo_math_divrot.cpp.f90:4404
2.967 ms	!$omp do @mo_advection_utils.cpp.f90:618
2.932 ms	!$omp do @mo_solve_nonhydro.cpp.f90:5225
2.816 ms	!$omp do @mo_solve_nonhydro.cpp.f90:4983
2.745 ms	__mo_communication_NMOD_exchange_data_r3d_
2.601 ms	!$omp do @mo_solve_nonhydro.cpp.f90:1083
2.45 ms	!$omp do @mo_solve_nonhydro.cpp.f90:1194
2.379 ms	!$omp do @mo_solve_nonhydro.cpp.f90:1112
2.193 ms	MPI_Barrier
2.149 ms	!$omp do @mo_solve_nonhydro.cpp.f90:4674
2.093 ms	!$omp do @mo_solve_nonhydro.cpp.f90:4587

Property	Value
Display	Process Timeline
Type	Function
Function	__mo_solve_nonhydro_NMOD_solve_nh_dynamic_
Function Group	Application
Interval Begin	6.905004 s
Interval End	6.997314 s
Duration	92.310022 ms
Source Location	mo_solve_nonhydro.cpp.f90:4473

Function Legend

🟩	Application
🟥	MPI
🟧	OMP
🟫	OMP-LOOP
🟩	OMP-PREG
🟦	OMP-SYNC

nproma=4, nchunksize=1, dynamic schedule

Program Analysis and Tuning Workshop - DKRZ 2012

Jörg Behrens
Deutsches Klimarechenzentrum GmbH

Panagiotis Adamidis, Hendryk Bockelman,
Thomas Jahns (DKRZ)

Example: ECHAM performance

In the ScalES project we looked at how model performance scales with the number of tasks.

Of course, the worst part must be the serial part, e.g. serial IO.

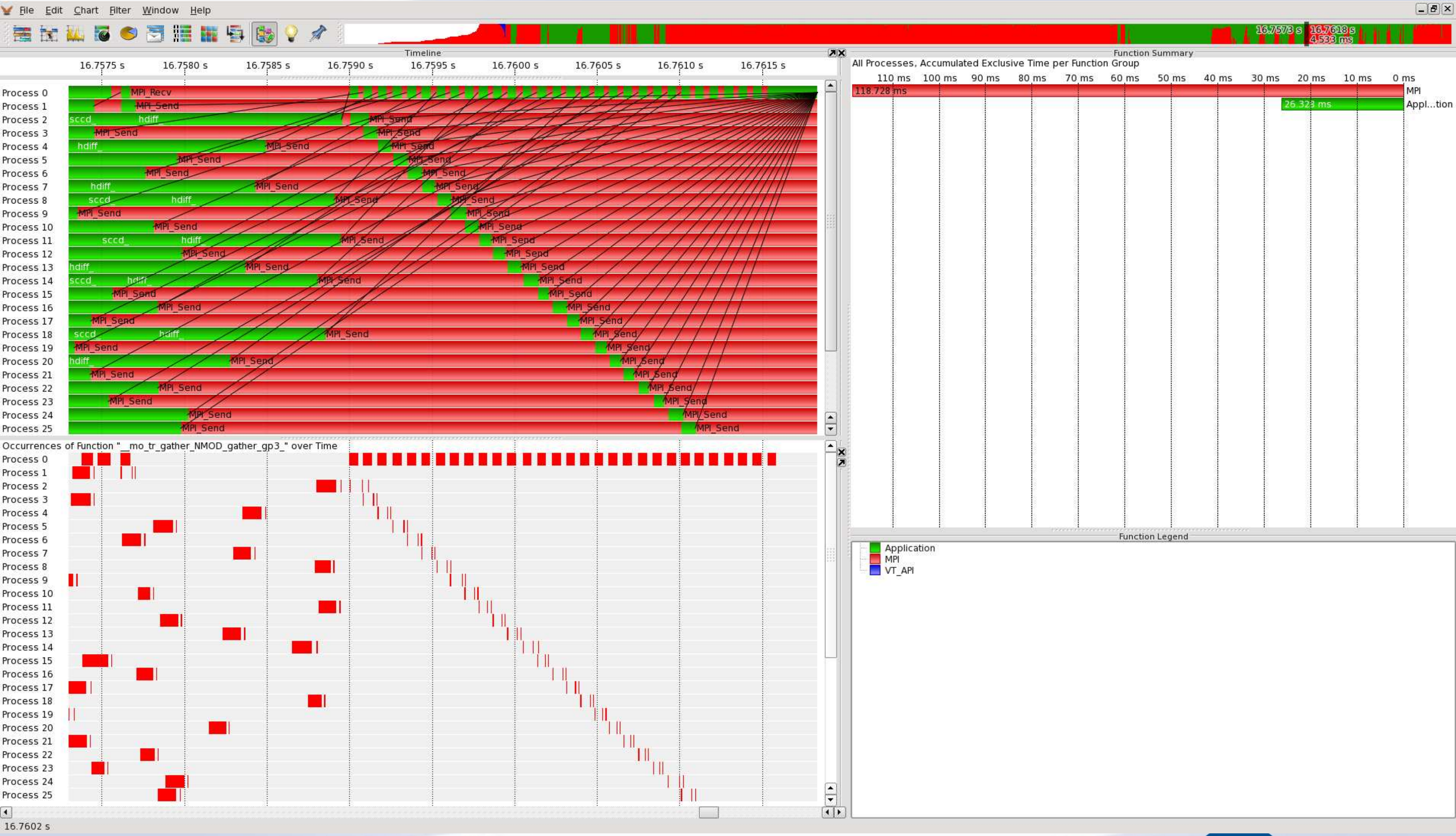
But it still can be worse than that:

identified bottleneck: hand-made p2p-gather

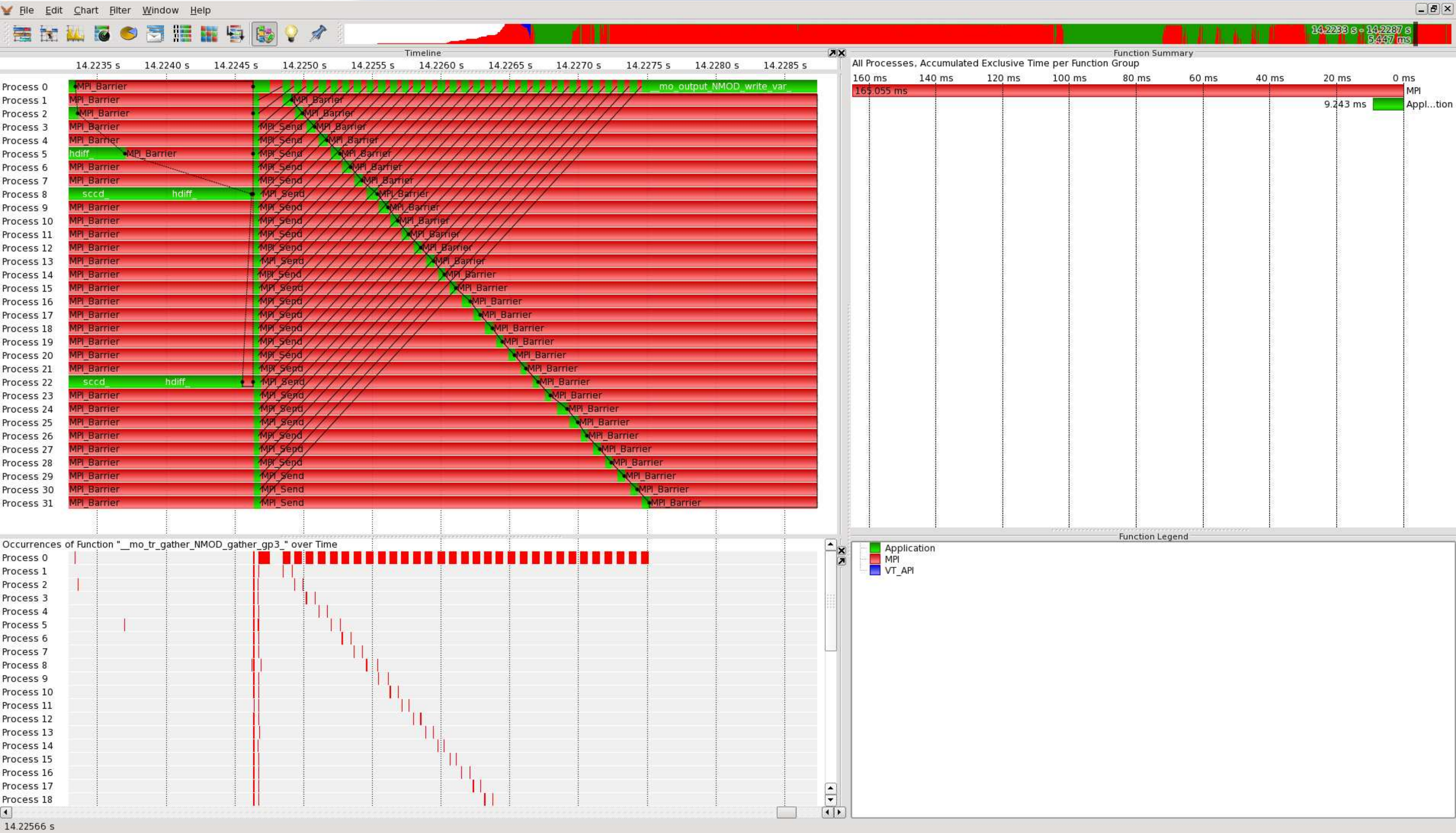
becomes slower for #task > 512 (ECHAM T127L95)

p2p-gather required because the MPI_GATHERV is not general enough

Vampir view at p2p-gather



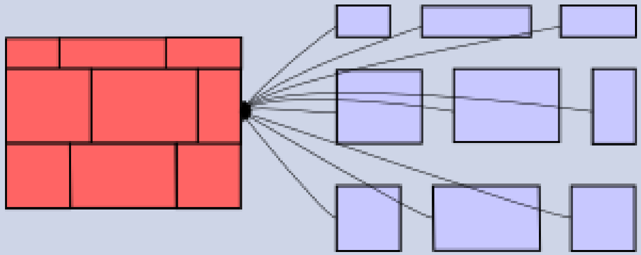
Insertion of barriers gives a compact context free view



New gather communication

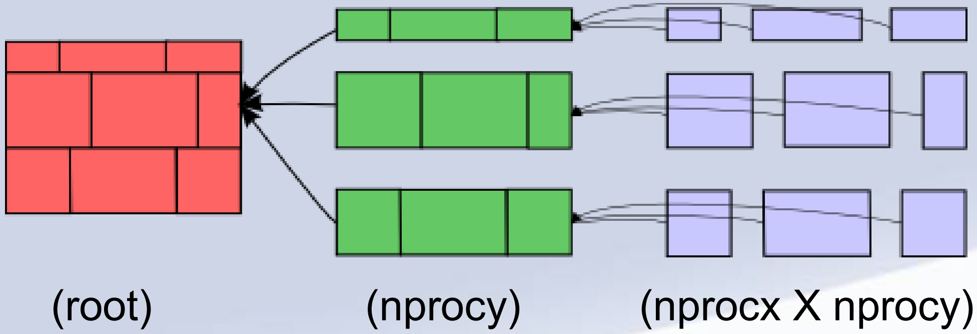
Problem:

- non-uniform but latitude-aligned subarrays
- collective MPI_GATHERV unapplicable



Old:

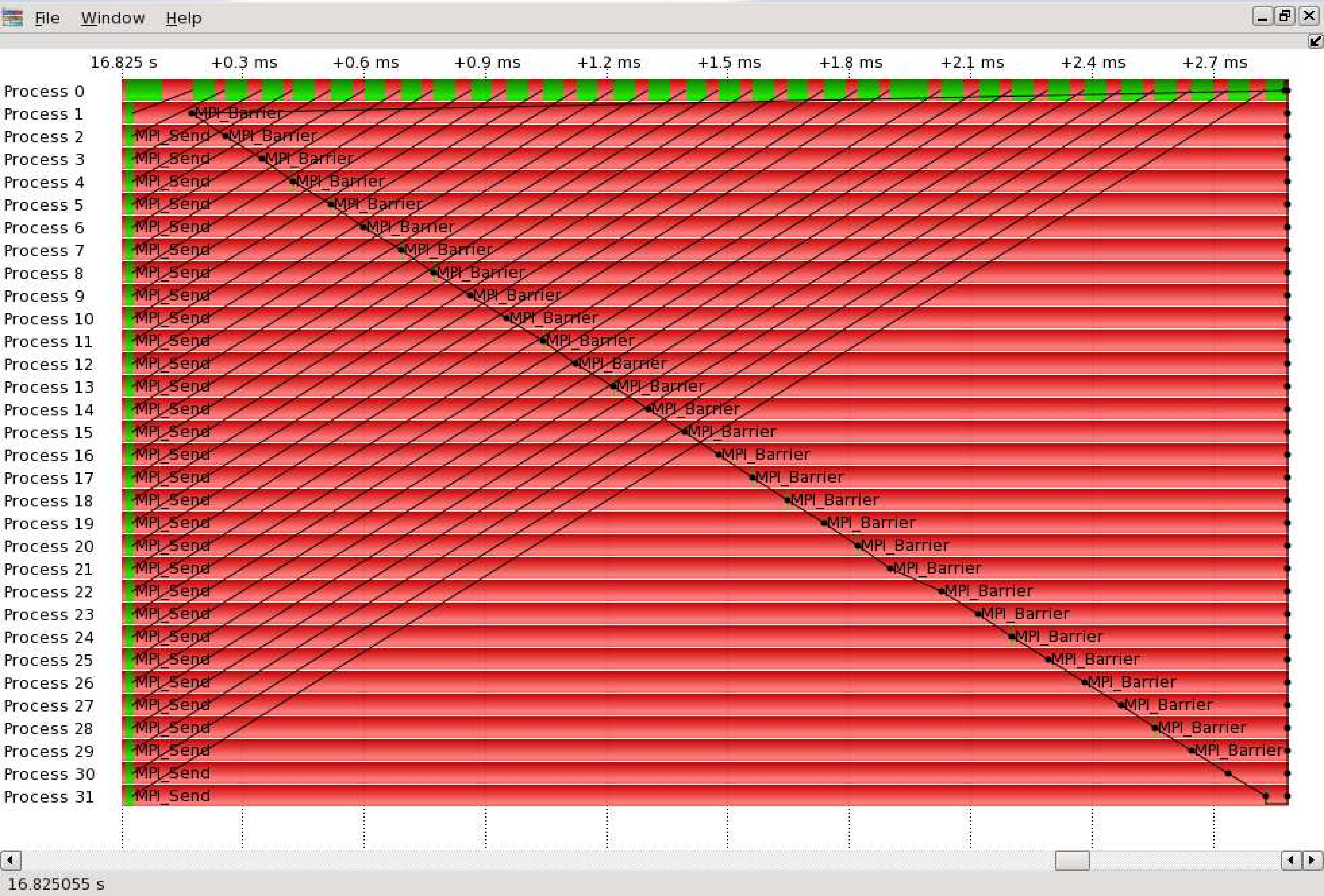
- many global p2p communications
- overhead concentrated on root
- increasing cost for high #tasks



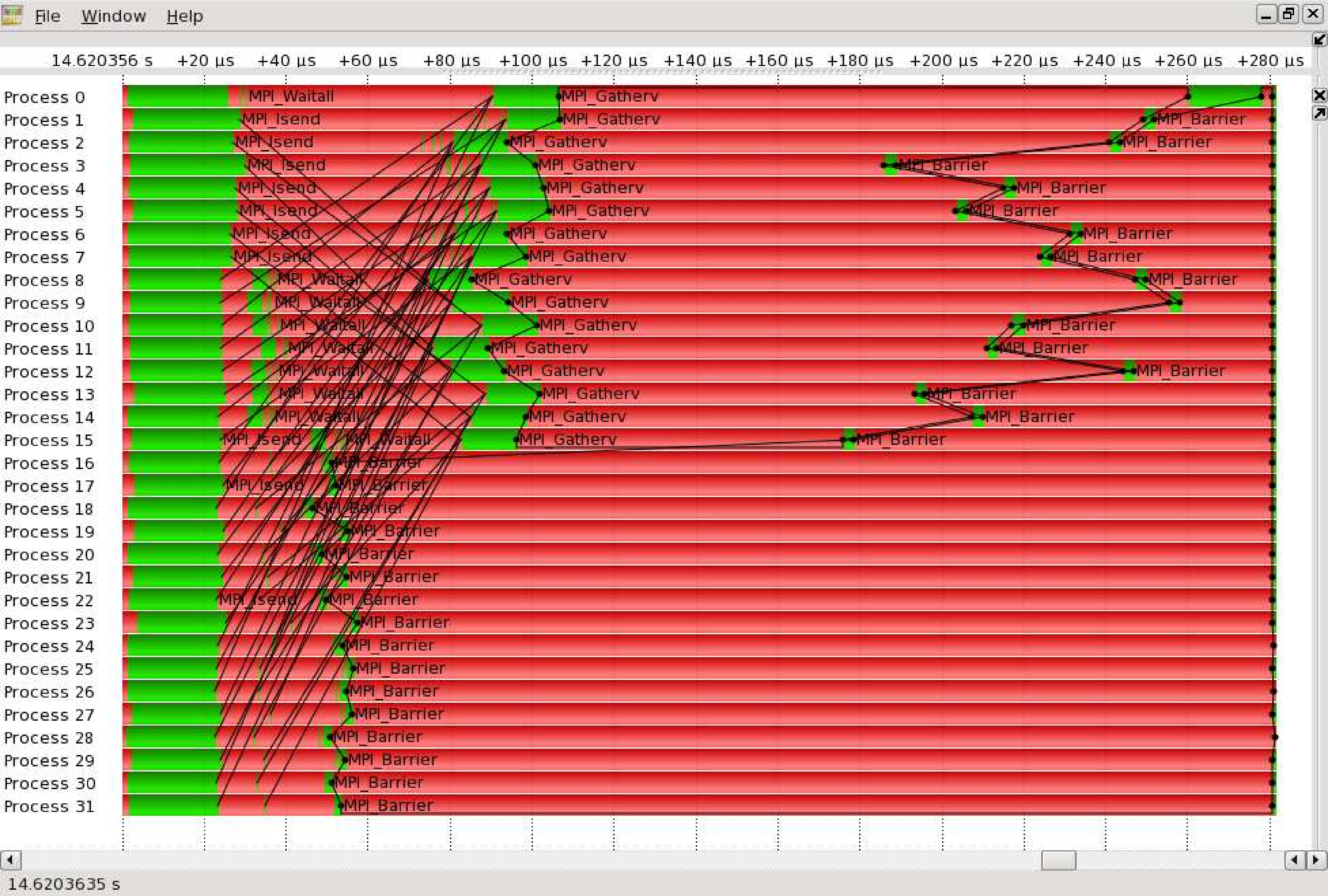
New:

- build latitude-aligned subgroups
- distribute shape-overhead
- fast collective MPI_GATHERV for second phase applicable
- const. cost for high #tasks

Old gather: N → 1

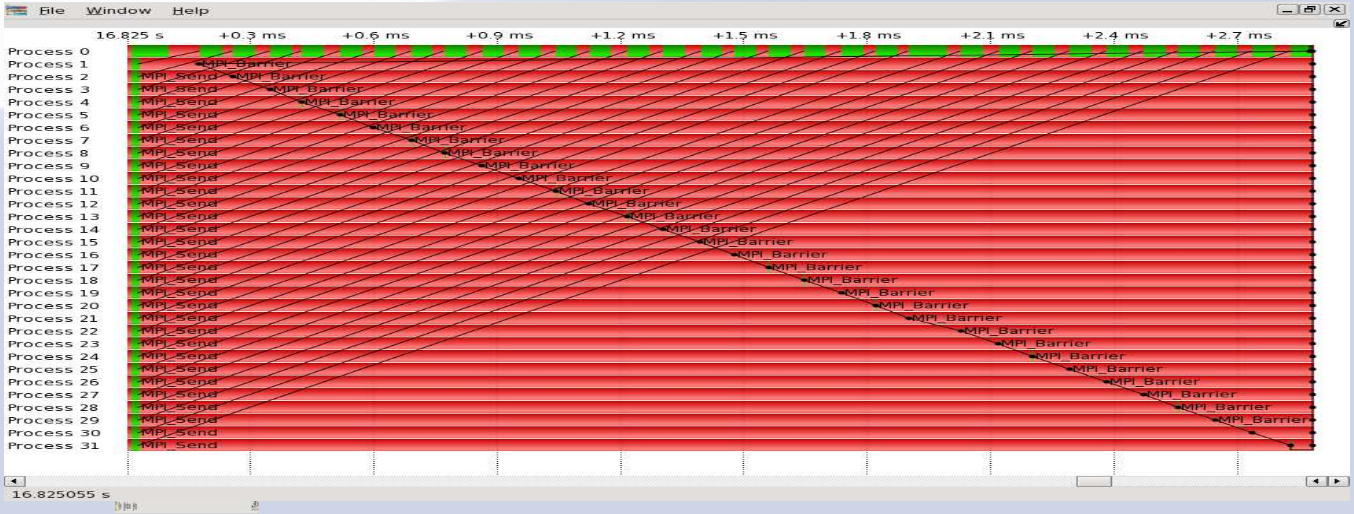


New two-phase gather: N → M → 1

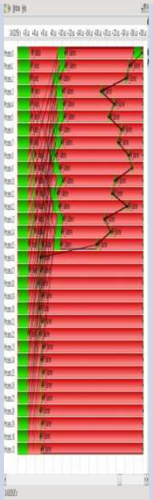


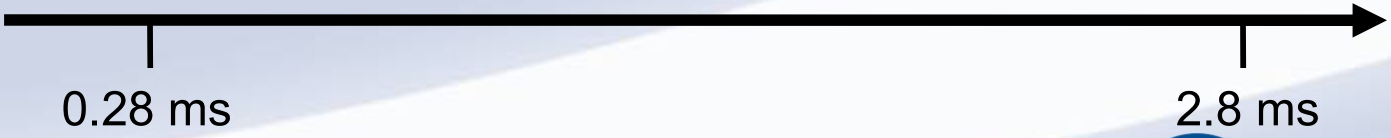
Gather: one-phase vs. two-phase

<u>Old:</u>
one-phase



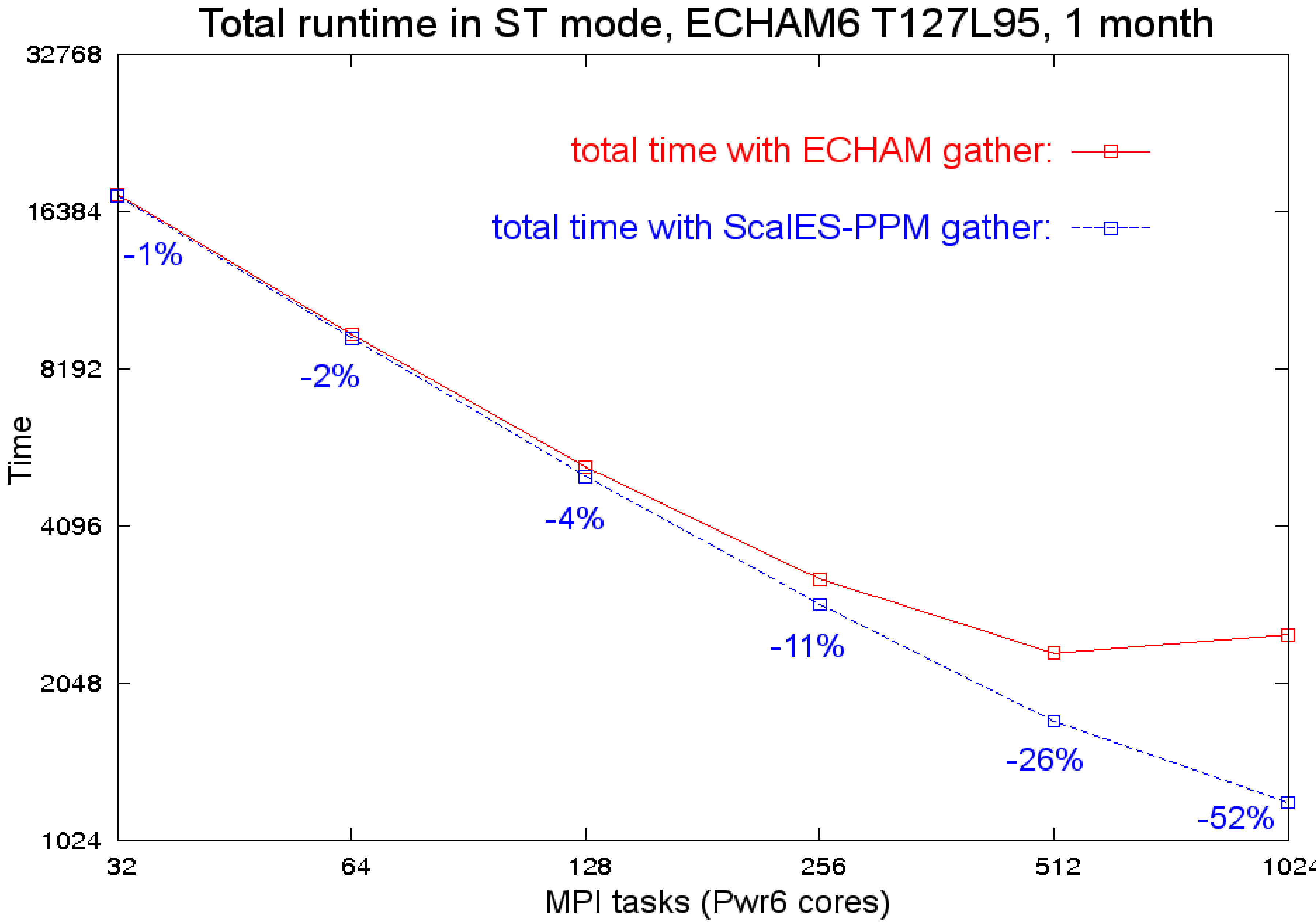
<u>New:</u>
two-phase





ScalES

Total runtime measurement



Example: MPIOM
Aggregation of communication

Major communication in MPIOM: boundsexchange

Old implementation:

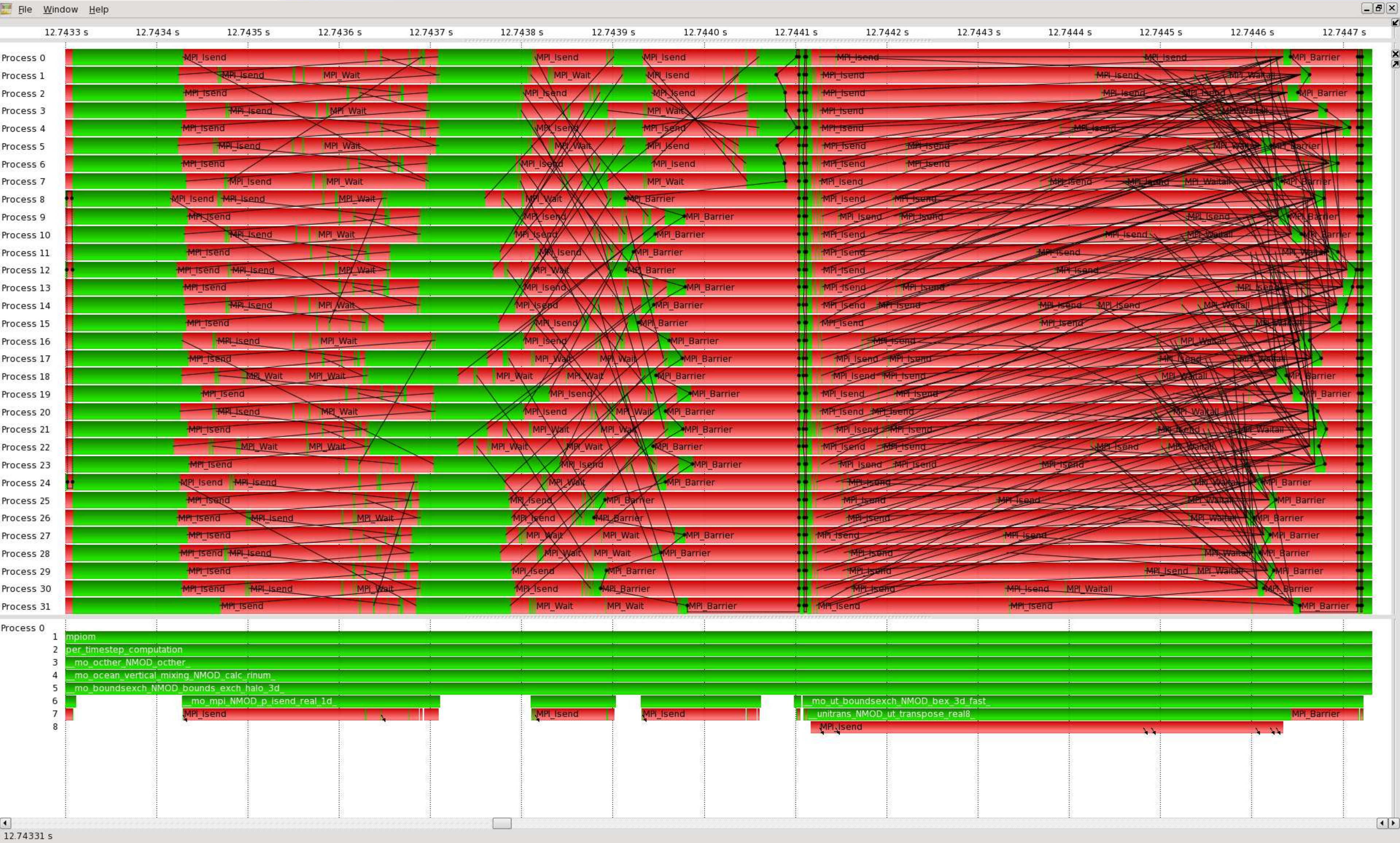
- Separated updates of x, y, special northborder
- User-buffered messages within each phase

New implementation:

- reprogram communication using a more abstract formulation (using the *Unitrans* communication library)
- *Unitrans* uses MPI datatypes

Vampir is used to document changes

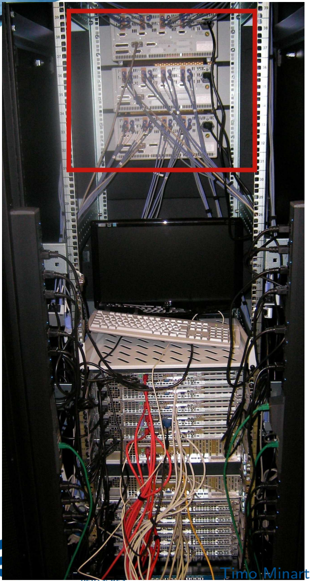
Comparison: old (left), new (right)



Working with vampirtrace plugin counters

Timo Minartz, DKRZ

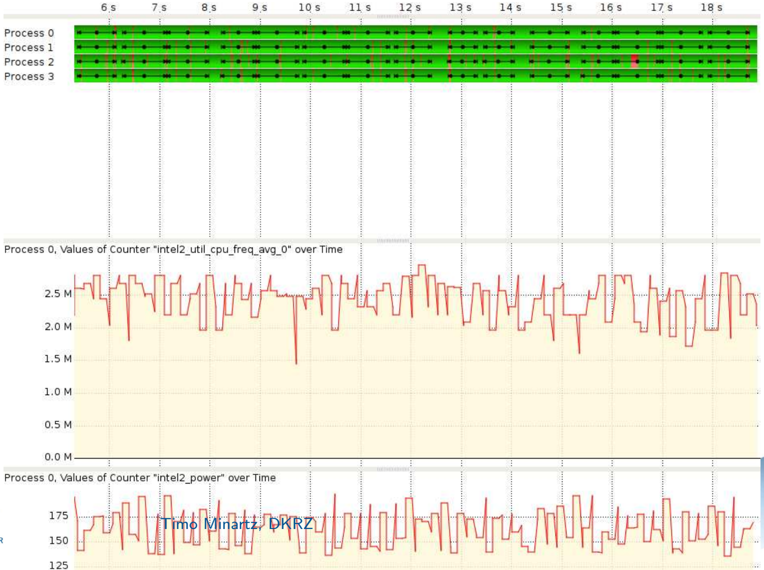
Correlating energy relevant metrics



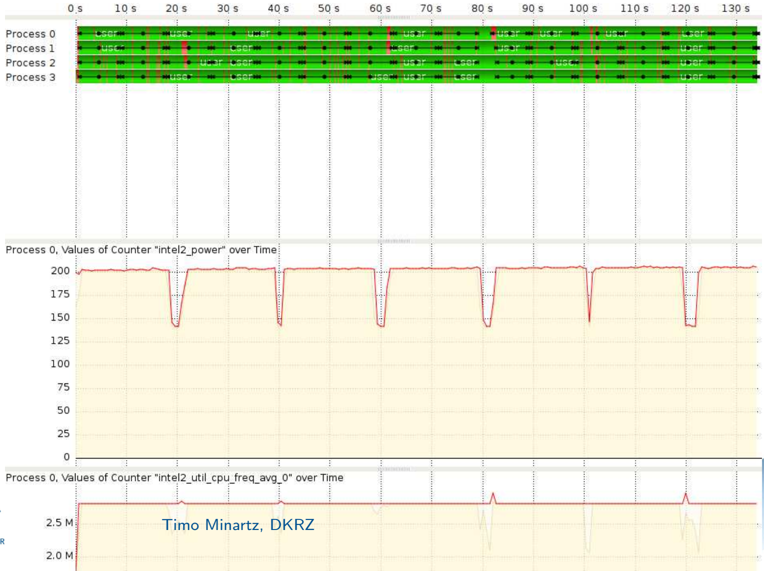
Idea

- Trace energy relevant metrics in database
  - Processor load, Performance Counter, . . .
  - Processor frequency, . . .
  - Power consumption
- Merge metrics post-mortem via VT Plugin Interface
- Switch processor frequency based on application phases

GETM model power variations



GETM model instrumented



Experiences and open questions

VT Plugin Interface

- Limitation to 256 counters
- Post-mortem integration takes a multiple of the application runtime...
- ... and crashes sometimes

```
[0]VampirTrace: FATAL: OTF_WStream_writeCounter failed:
 ERROR in function OTF_WBuffer_setTimeAndProcess,
 file: ../../../../extlib/otf/otflib/OTF_WBuffer.c, line: 296:
 time not increasing. (t= 14835708593910, p= 1)
```