ScalaLife

Scalable Software Services for Life Science

Codes performance analysis

Judit Gimenez - BSC

DISCLAIMER:

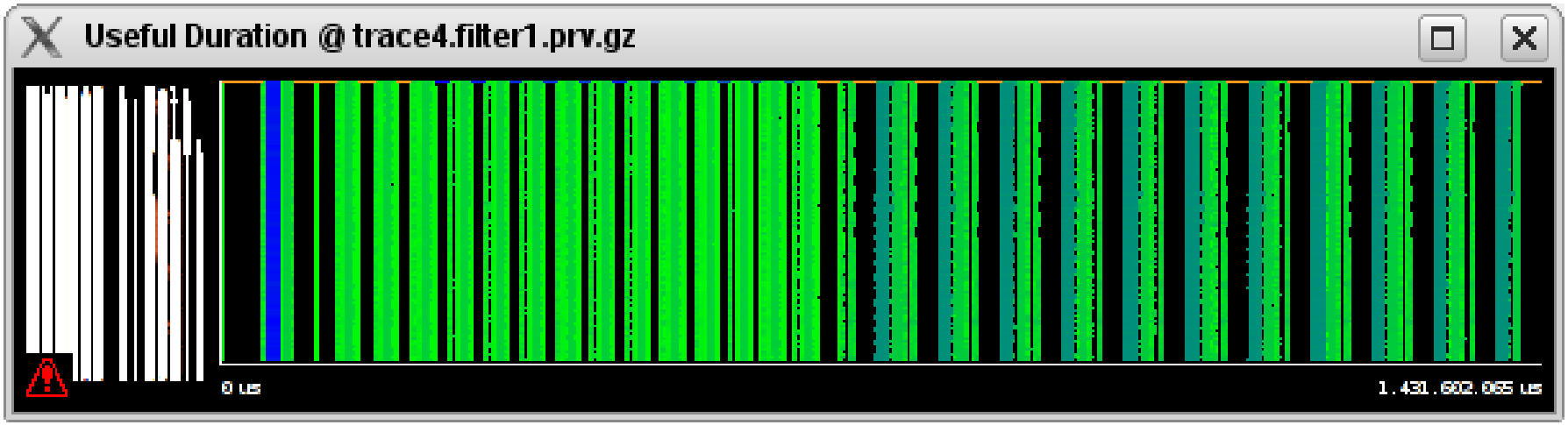
This is a almost blind analysis. I have not looked at the applications source code and I have no knowledge of what they compute...

DALTON @ Lindgren

Code structure

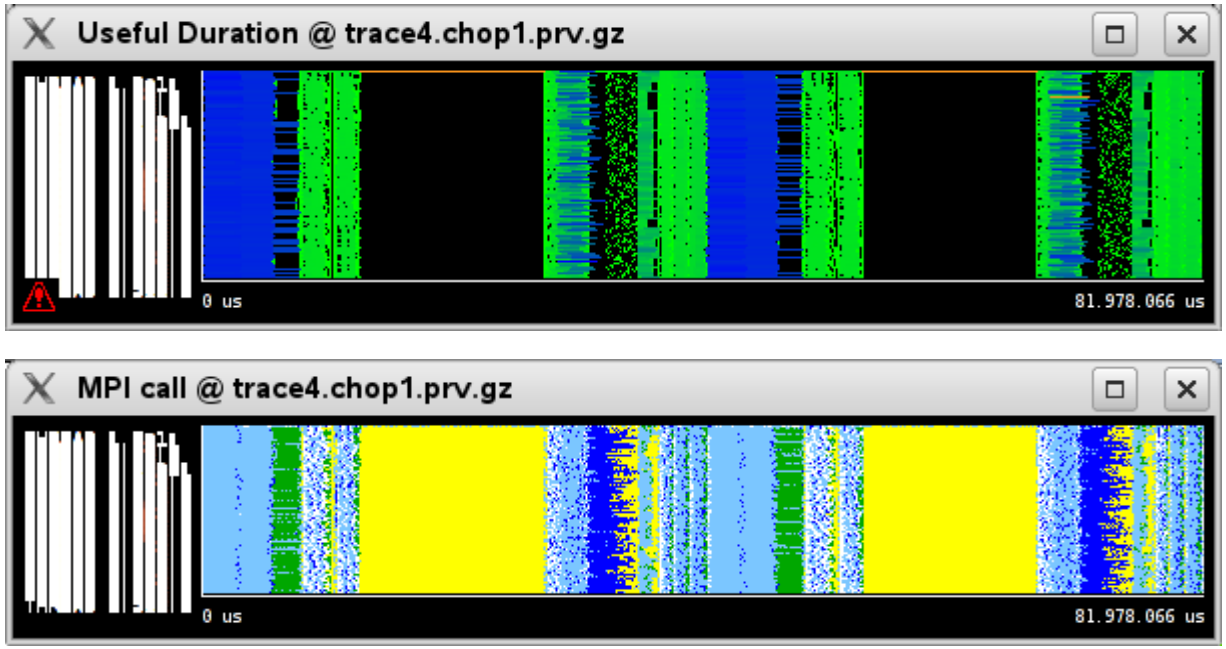
- 512 tasks, relevant input case?
- Two phases

wave function properties



Wave function

- 2 iterations
- Dominated by MPI



MPI_Bcast 15 sec

MPI _Reduce (unbalance)

serialization eliminated

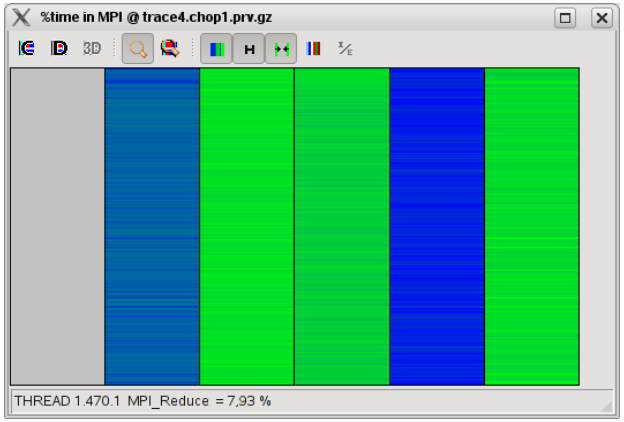
Wave function: MPI

Scalable Software Services for Life Science

%time in MPI @ trace4.chop1.prv.gz

	End	MPI_Send	MPI_Recv	MPI_Bcast	MPI_Reduce
THREAD 1.1.1	43,99 %	17,63 %	34,03 %	1,17 %	3,18 %
THREAD 1.2.1	41,04 %	6,01 %	8,26 %	42,63 %	2,06 %
THREAD 1.3.1	38,04 %	7,56 %	8,33 %	40,14 %	5,93 %
THREAD 1.508.1	35,81 %	6,28 %	9,57 %	45,73 %	2,61 %
THREAD 1.509.1	35,70 %	7,41 %	9,53 %	44,47 %	2,90 %
THREAD 1.510.1	34,19 %	4,91 %	8,92 %	47,30 %	4,68 %
THREAD 1.511.1	40,08 %	6,17 %	7,88 %	44,53 %	1,34 %
THREAD 1.512.1	38,15 %	6,86 %	7,73 %	45,09 %	2,16 %
Total	16.111,70 %	3.632,83 %	5.355,73 %	22.068,06 %	4.031,68 %
Average	31,47 %	7,10 %	10,46 %	43,10 %	7,87 %
Maximum	43,99 %	17,63 %	34,03 %	48,28 %	12,30 %
Minimum	27,48 %	4,31 %	7,23 %	1,17 %	1,34 %
StDev	2,27 %	1,18 %	1,61 %	2,55 %	1,63 %
Avg/Max	0,72	0,40	0,31	0,89	0,64

THREAD 1.96.1



Two iterations only

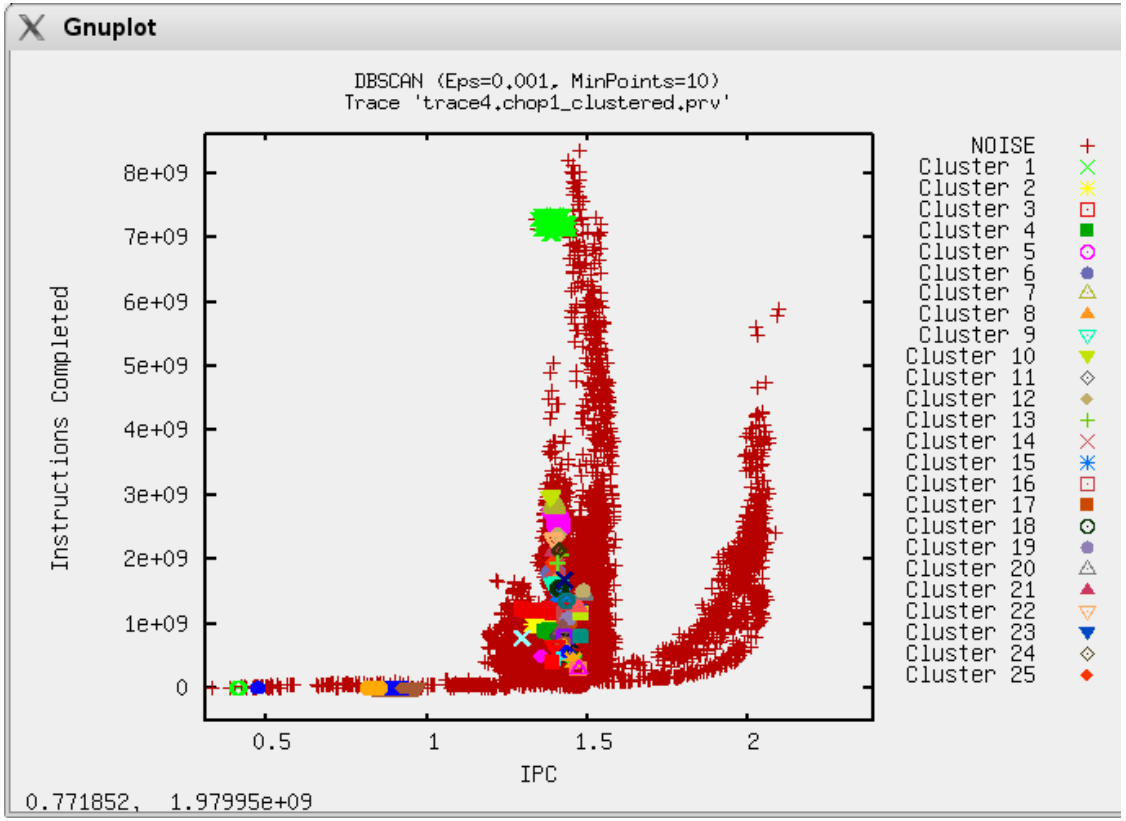
number of MPI calls @ trace4.chop1.prv.gz

	MPI_Send	MPI_Recv	MPI_Bcast	MPI_Reduce
THREAD 1.1.1	39.518	30.742	330	32
THREAD 1.2.1	70	92	334	32
THREAD 1.3.1	72	94	334	32
THREAD 1.4.1	70	94	334	32
THREAD 1.5.1	62	79	334	32
THREAD 1.6.1	62	81	334	32
THREAD 1.7.1	61	77	334	32
THREAD 1.8.1	62	78	334	32
THREAD 1.9.1	63	81	334	32
THREAD 1.10.1	62	79	334	32

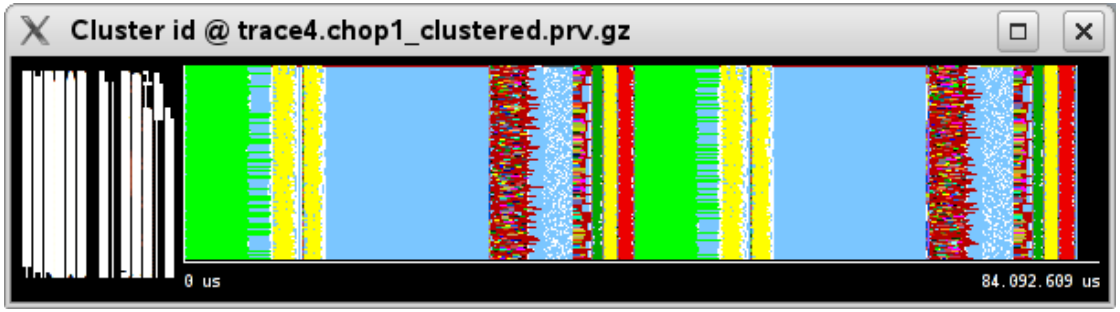
THREAD 1.5.1 End = 34,30 %

On average only 31% time computing
Maximum time out of MPI 44%

Wave f.: Computation

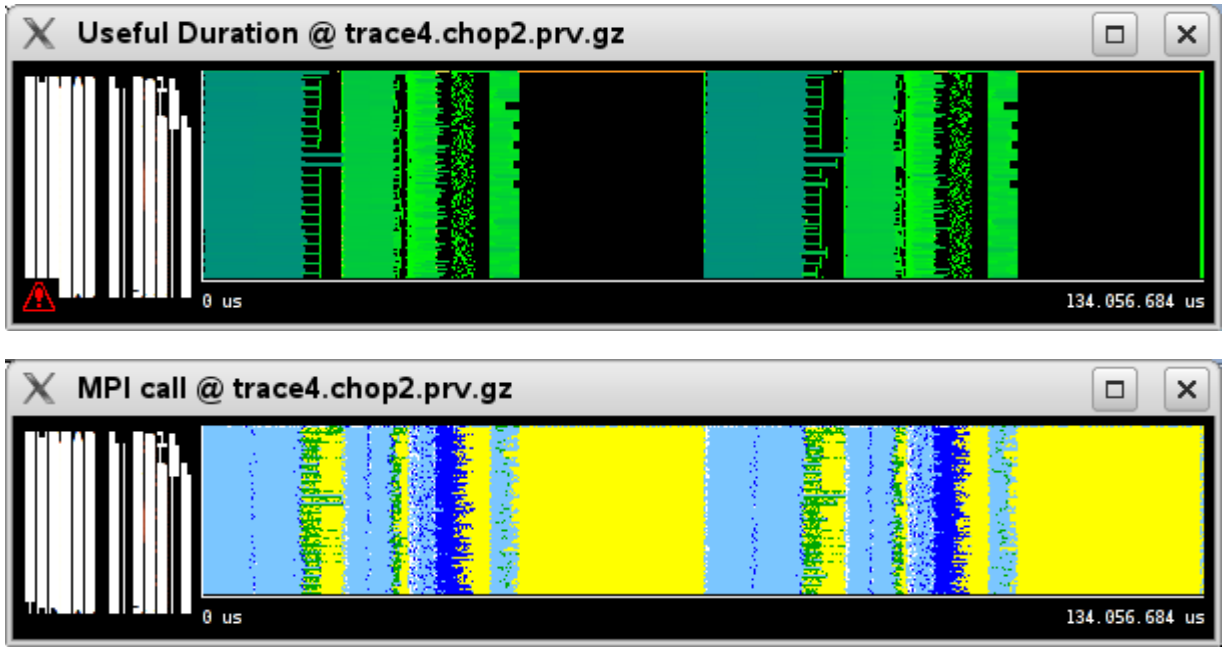


Main computation regions
- good performance
- balanced on instructions
- small variability on IPC



Properties

- 2 iterations
- Also dominated by MPI



MPI_Bcast
24.5 sec

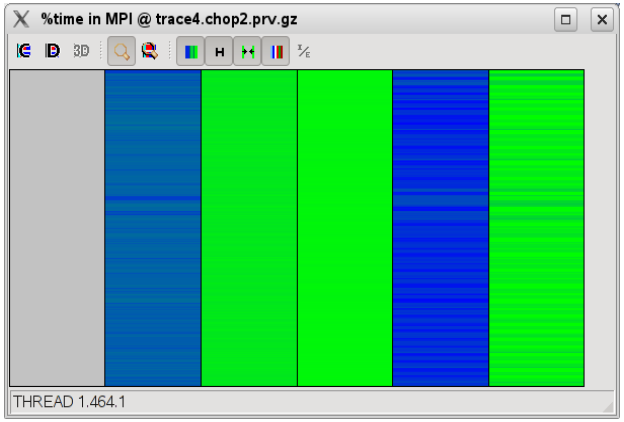
Properties: MPI

Scalable Software Services for Life Science

%time in MPI @ trace4.chop2.prv.gz

	End	MPI_Send	MPI_Recv	MPI_Bcast	MPI_Reduce
THREAD 1.1.1	46,04 %	6,72 %	42,29 %	0,40 %	4,54 %
THREAD 1.2.1	46,58 %	6,52 %	1,73 %	42,07 %	3,10 %
THREAD 1.3.1	45,37 %	4,70 %	2,12 %	44,07 %	3,74 %
THREAD 1.508.1	39,31 %	5,97 %	1,22 %	48,65 %	4,85 %
THREAD 1.509.1	38,24 %	6,02 %	1,61 %	48,98 %	5,15 %
THREAD 1.510.1	41,10 %	5,23 %	1,47 %	47,46 %	4,74 %
THREAD 1.511.1	43,91 %	4,66 %	0,74 %	50,51 %	0,18 %
THREAD 1.512.1	43,74 %	5,49 %	1,18 %	49,58 %	0,02 %
Total	19.524,81 %	2.738,35 %	1.156,22 %	25.783,56 %	1.997,06 %
Average	38,13 %	5,35 %	2,26 %	50,36 %	3,90 %
Maximum	46,63 %	6,74 %	42,29 %	59,23 %	14,92 %
Minimum	34,38 %	3,64 %	0,74 %	0,40 %	0,01 %
StDev	2,18 %	0,69 %	1,80 %	4,37 %	3,26 %
Avg/Max	0,82	0,79	0,05	0,85	0,26

End



A little bit better parallel efficiency but with worst communication efficiency

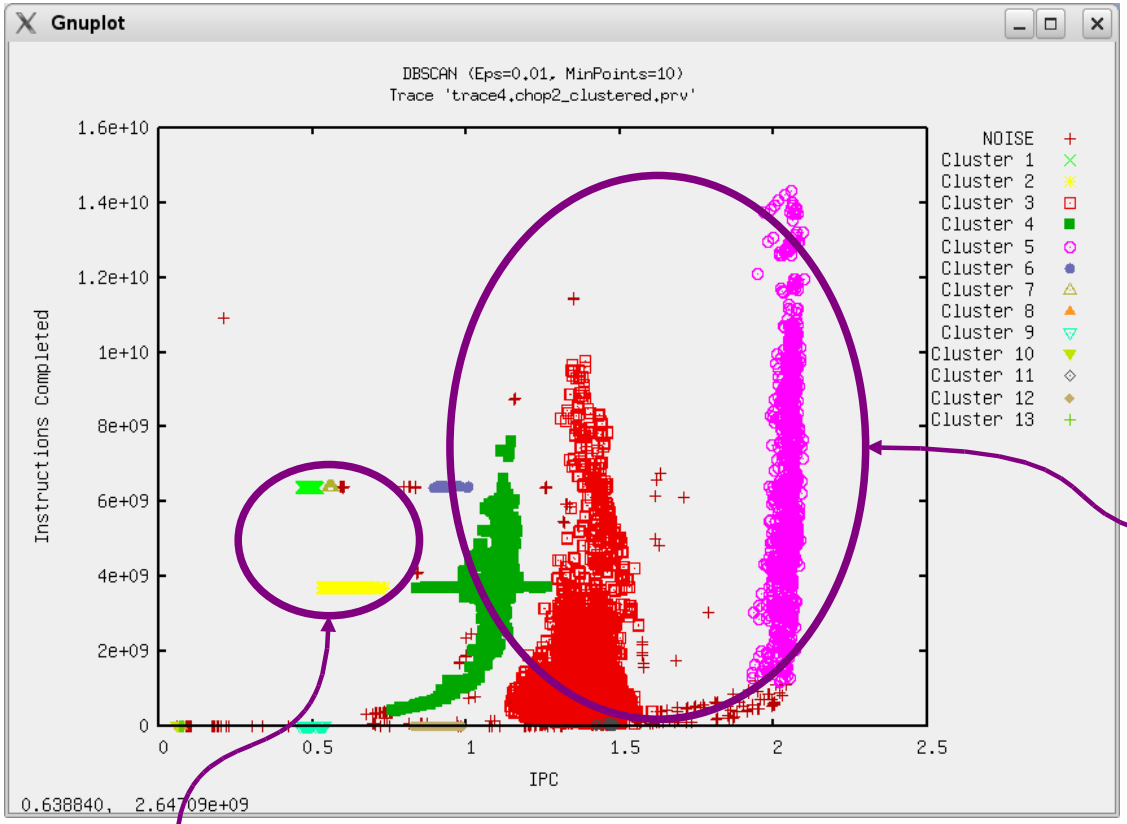
Half of the time waiting for the master

number of MPI calls @ trace4.chop2.prv.gz

	MPI_Send	MPI_Recv	MPI_Bcast	MPI_Reduce
THREAD 1.1.1	26.750	17.973	186	6
THREAD 1.2.1	42	66	190	6
THREAD 1.3.1	41	64	190	6
THREAD 1.4.1	42	66	190	6
THREAD 1.5.1	43	68	190	6
THREAD 1.6.1	40	62	190	6
THREAD 1.7.1	31	44	190	6
THREAD 1.8.1	34	50	190	6
THREAD 1.9.1	39	60	190	6
THREAD 1.10.1	36	54	190	6

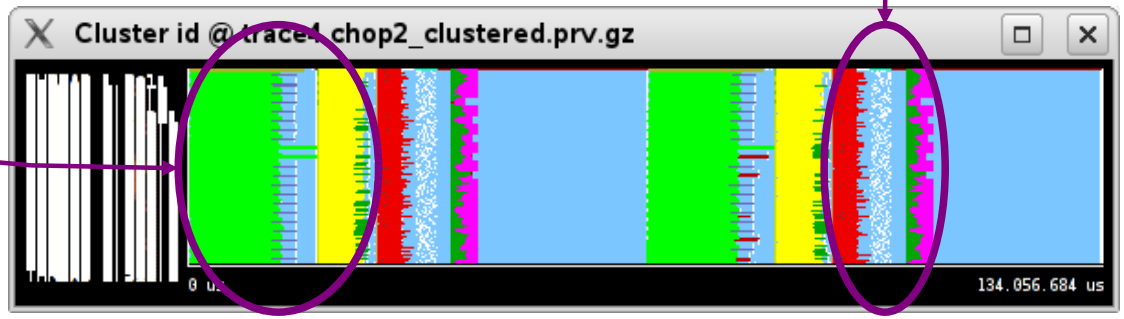
End

Properties: Computation



Huge variability on instructions

Most relevant computing regions obtain poor IPC



Properties: Computation

Scalable Software Services for Life Science

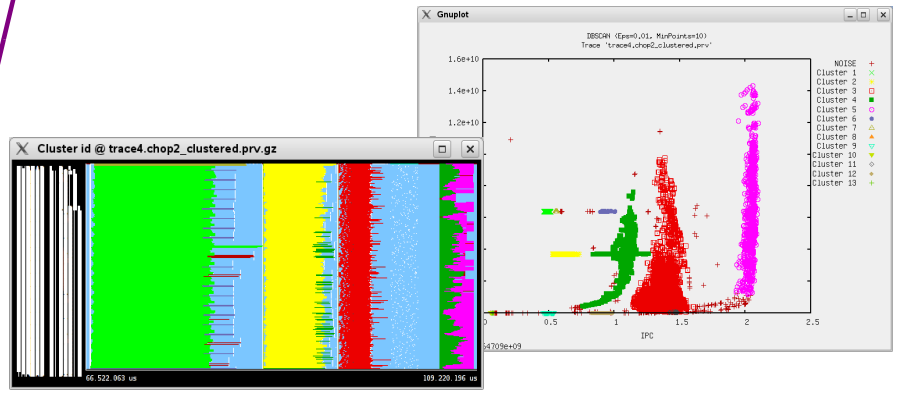
	C1	C2	C3	C4	C5	C4 line	C4 rest
ipc	0,48	0,59	1,37	1,09	2,05	1,02	1,09
l1/ins	0,19%	0,22%	0,94%	0,43%	0,01%	0,24%	0,45%
l2/ins	0,14%	0,15%	0,09%	0,22%	0,00%	0,18%	0,22%
tlb/ins	0,04%	0,04%	0,01%	0,07%	0,00%	0,04%	0,07%
fpi/ins	33,13%	34,23%	19,33%	34,99%	0,01%	34,23%	34,89%
fpo/ins	64,61%	66,23%	22,06%	68,65%	0,01%	66,21%	68,74%
fdv/fpi	**0,12%**	**0,19%**	0,77%	**0,03%**	0,00%	**0,19%**	**0,01%**
fml/fpi	**72,22%**	**63,69%**	60,71%	**57,41%**	0,00%	**63,70%**	**57,18%**
vec/ins	74,84%	67,39%	45,40%	71,34%	0,04%	67,38%	72,21%
br/ins	5,97%	6,73%	9,38%	7,11%	20,38%	6,73%	7,06%
fpu_idle/cyc	3,43%	7,04%	18,79%	6,79%	98,62%	**12,67%**	6,22%
res_stl/cyc	79,45%	73,89%	32,00%	54,40%	5,66%	57,64%	53,83%
no_ins/cyc	1,13%	2,21%	7,14%	2,10%	4,86%	4,13%	1,96%

What do C5 does?

Stalled on resources
No instruction issued

C1/C2 similar/better to C4 except fdiv/fml ratio

C4 line equivalent to C2 except fpu_idle ratio

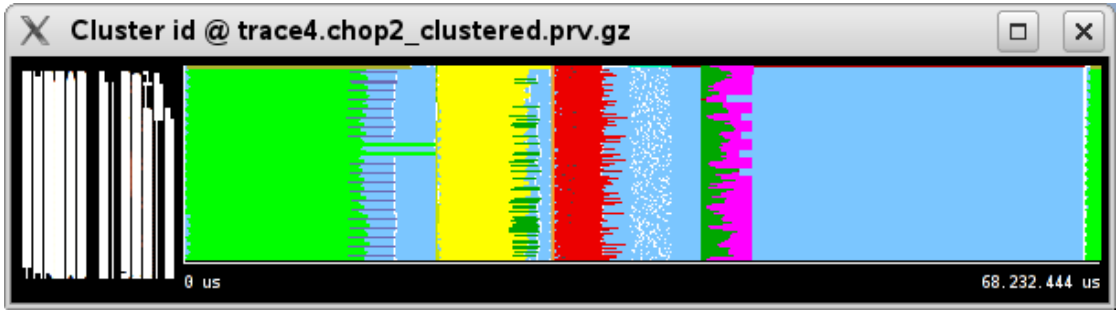


ScalaLife

Code modifications

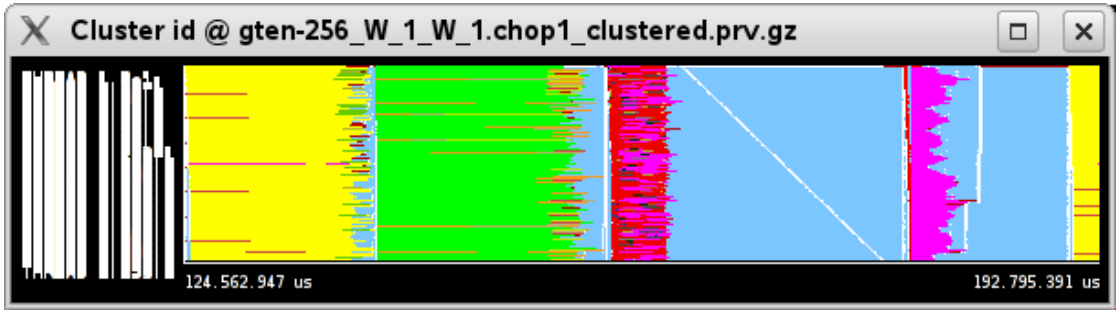
- Some problems reading callers @ lindgren
- Tried to correlate with previous trace of 256 tasks (before eliminating serialization) but seems that more changes have been implemented or a different configuration…

512 @ lindgren



Properties iteration
@ same scale

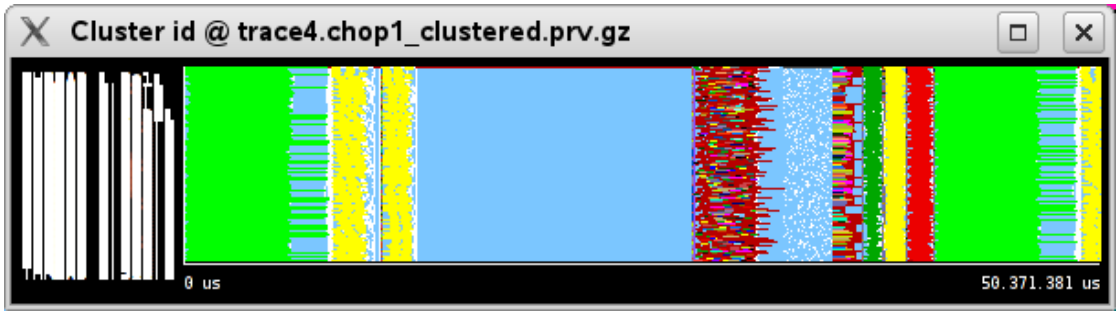
256 @ other
KTH machine



Code modifications

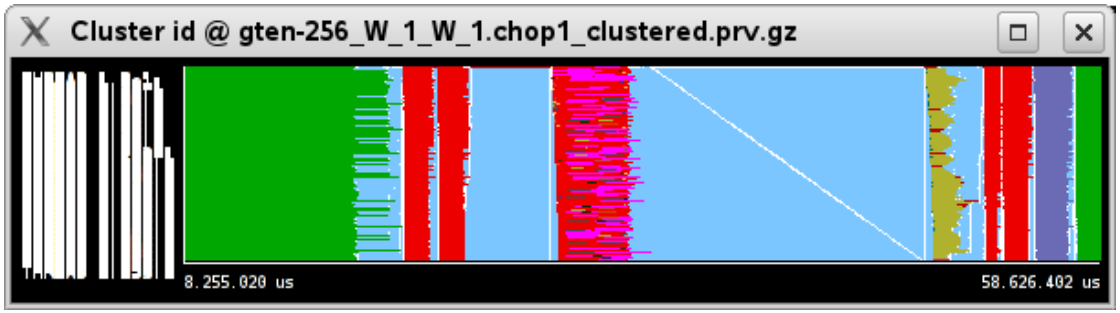
- Bigger improvements on the wave function phase?

512 @ lindgren



Wave function iteration @ same scale

256 @ other KTH machine



DALTON conclusions

- Reduce master bottleneck
  - Large sequential bursts (not so large on previous version?) – I/O? depends on #task, configuration?
  - Point 2 point communications: reduce number? avoid forcing any order
- Unbalance is the second problem
  - Parallelise last chunk reusing available CPUs waiting on MPI – SMPSs + load balance
- Poor IPC on properties main computing regions
  - What are the differences between *cluster4-line* and cluster 2? May even be the same source code.
  - The clue may be on the distribution of the expensive FPO (div, sqrt, mul?)

DISCRETE @ MN

Code structure

- Sequential code, relevant input case?
- Latest version from early this week
  - Traces obtained Wednesday evening
  - Analysis mainly during All hands session (sorry!)
- Completely rewritten from last analysis
  - Previously 2 main user functions (potencial & colisio)
  - Now 49 routines, what are the important ones?
- Even the input files have changed, so cannot easily compare with previous versions

gprof flat profile

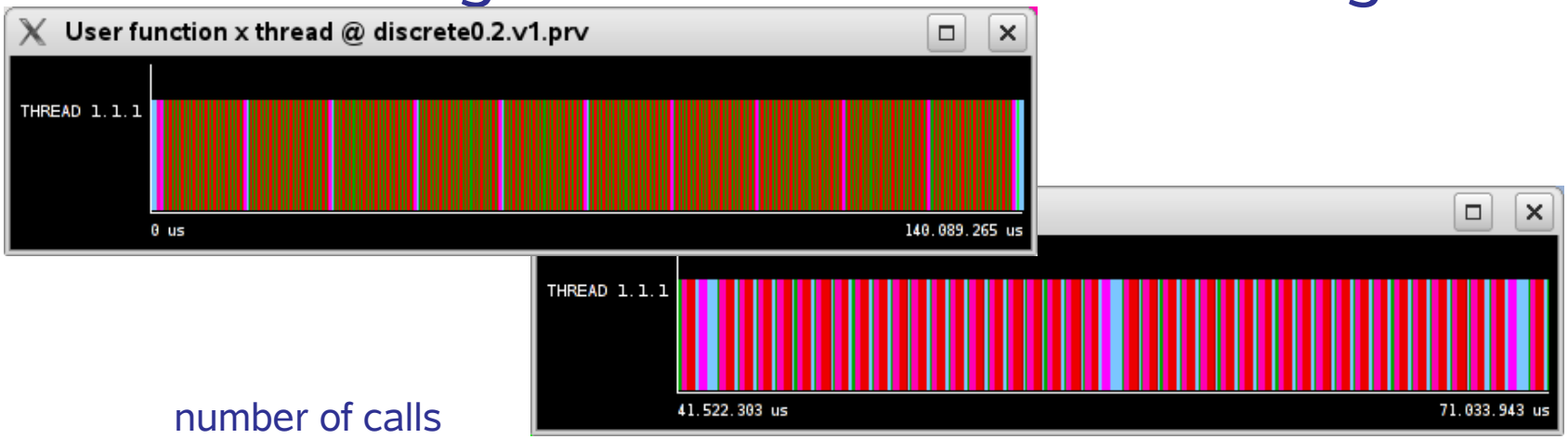
Scalable Software Services for Life Science

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
24.11	24.64	24.64	201	0.12	0.12	.calcekin_
17.84	42.87	18.23	201	0.09	0.09	.activatesteppot_
16.93	60.18	17.30				.getmintcol_
15.22	75.74	15.56	791777531	0.00	0.00	.__paramset__readinputparamset
9.30	85.25	9.51	1	9.51	58.48	.MAIN__
5.10	90.46	5.21	44652538	0.00	0.00	.updatetcol_
2.98	93.51	3.05	1416468	0.00	0.00	.nextcol_
2.64	96.21	2.70				.chgmom_
2.40	98.66	2.45				.calccm_
0.98	99.66	1.00	101	0.01	0.14	.thermalize_
0.71	100.39	0.73	12949192	0.00	0.00	.inici_
0.58	100.98	0.59	200	0.00	0.00	.prepintlist_
0.53	101.52	0.54				.colisiobond_
0.20	101.72	0.20	490544	0.00	0.00	.colisiononbond_
0.19	101.91	0.19	200	0.00	0.01	.__commline__writehelptext

(...)

First tracefile

- First tracefile 166MB (with getmintcol 540MB)
  - Some fine grain routines within the orange



number of calls

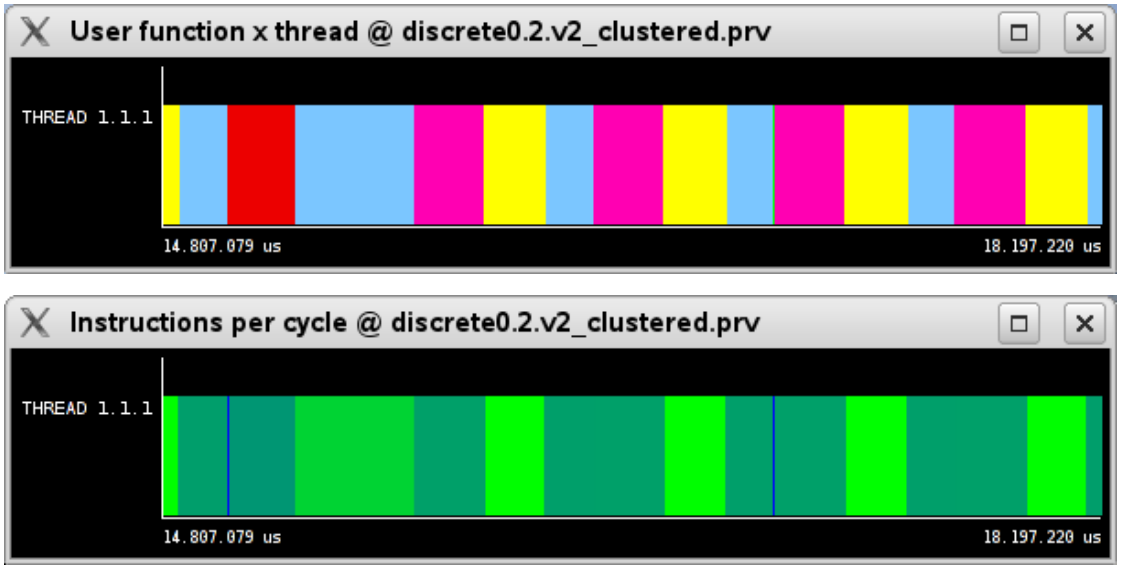
calcekin_	201
activatesteppot_	201
chgmom_	490.344
calccm_	12
thermalize_	202
prepintlist_	200
colisiobond_	490.475
calcepot_	11

%time

calcekin_	0,00 %
activatesteppot_	52,65 %
chgmom_	0,57 %
calccm_	0,00 %
thermalize_	0,07 %
prepintlist_	43,14 %
colisiobond_	0,90 %
calcepot_	2,66 %

Second tracefile

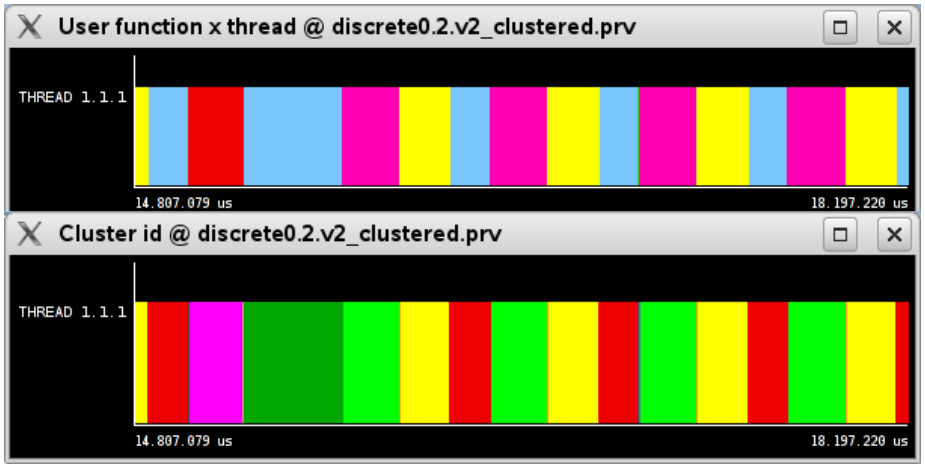
- Tracefile size 292 KB



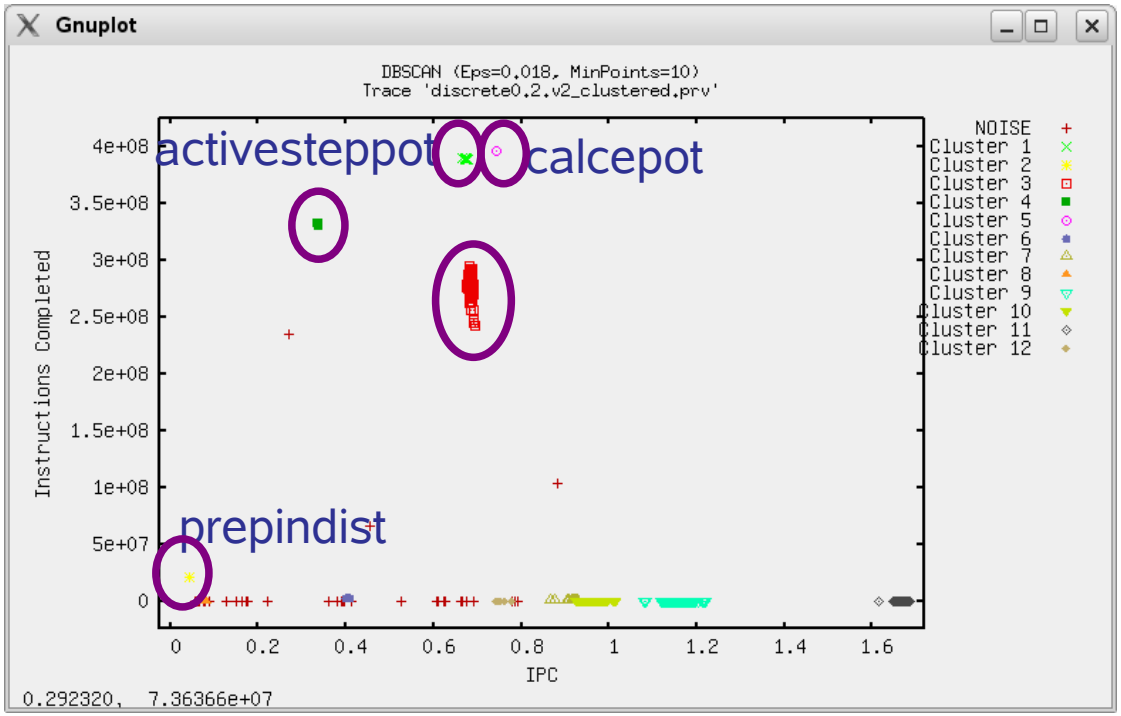
	%time	duration (avg)
End	30,25 %	67.773,67 us
calcekin_	0,00 %	22,17 us
activatesteppot_	36,50 %	254.697,79 us
calccm_	0,00 %	27,76 us
thermalize_	0,05 %	357,79 us
prepintlist_	31,34 %	219.795,04 us
calcepot_	1,85 %	235.346,57 us

Clustering computing

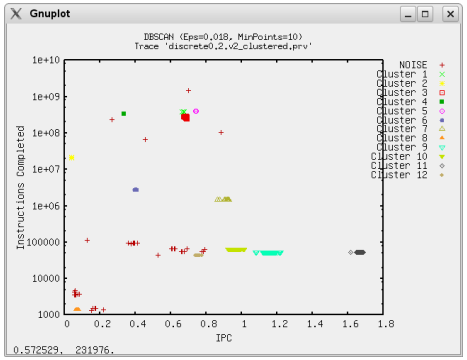
Scalable Software Services for Life Science



Cluster 1	33,96 %	activesteppot
Cluster 2	29,78 %	prepintlist
Cluster 3	24,23 %	
Cluster 4	3,17 %	calcepot
Cluster 5	1,68 %	
Cluster 6	0,19 %	
Cluster 7	0,05 %	thermalize
Cluster 8	0,00 %	
Cluster 9	0,00 %	calcekin
Cluster 10	0,00 %	
Cluster 11	0,00 %	
Cluster 12	0,00 %	

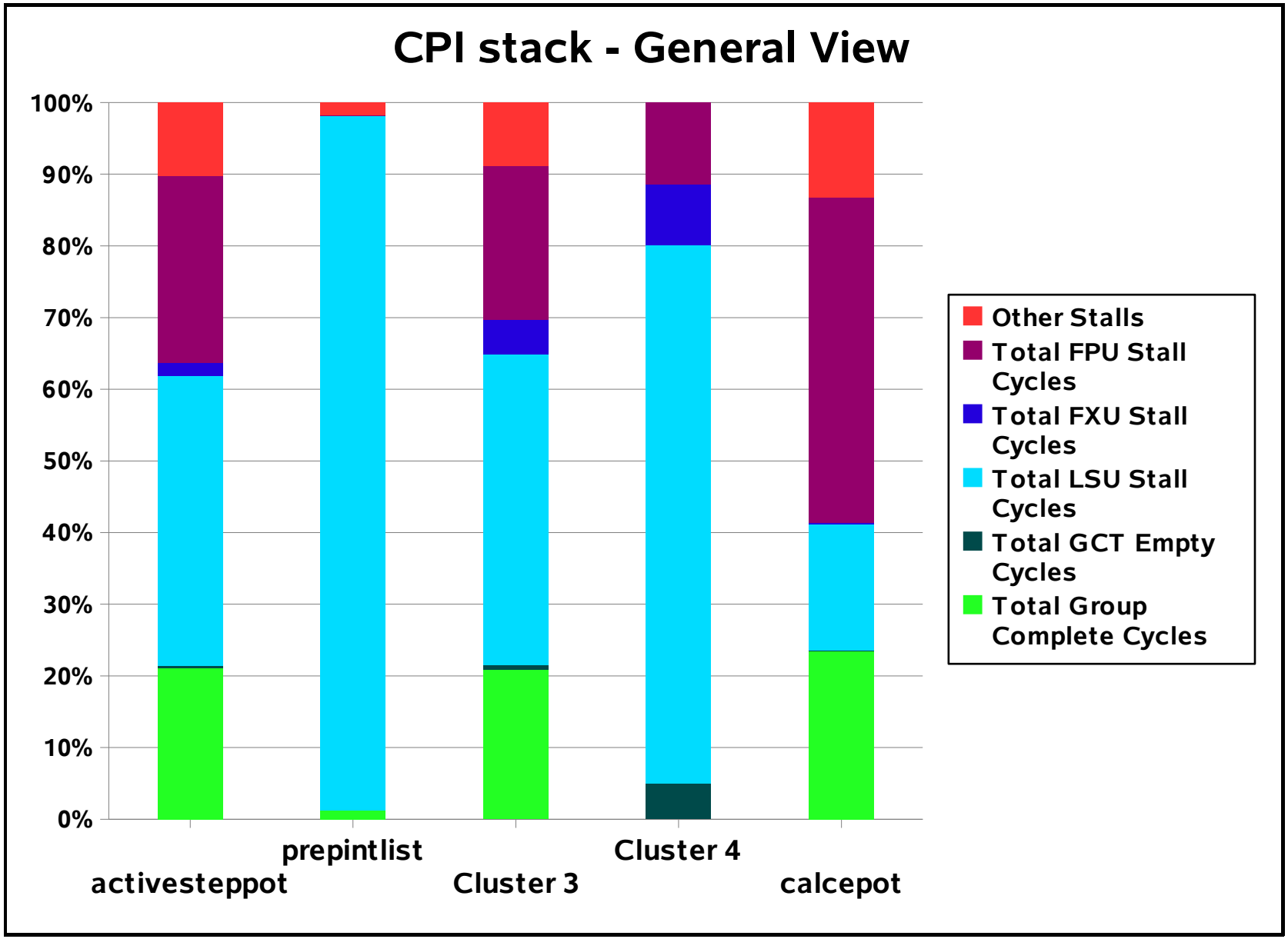


log scale



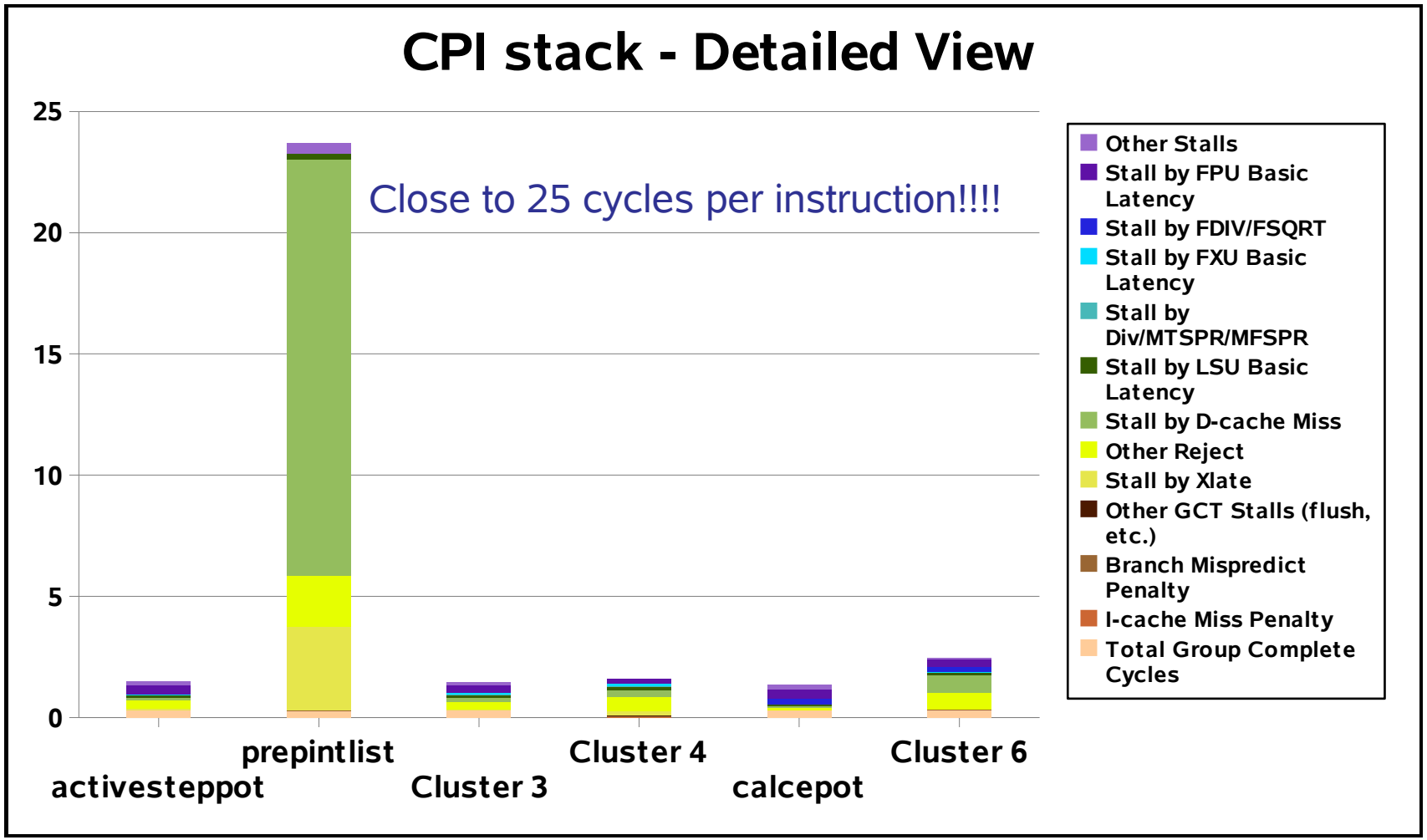
ScalaLife

CPIStack analysis



CPIStack analysis

Scalable Software Services for Life Science

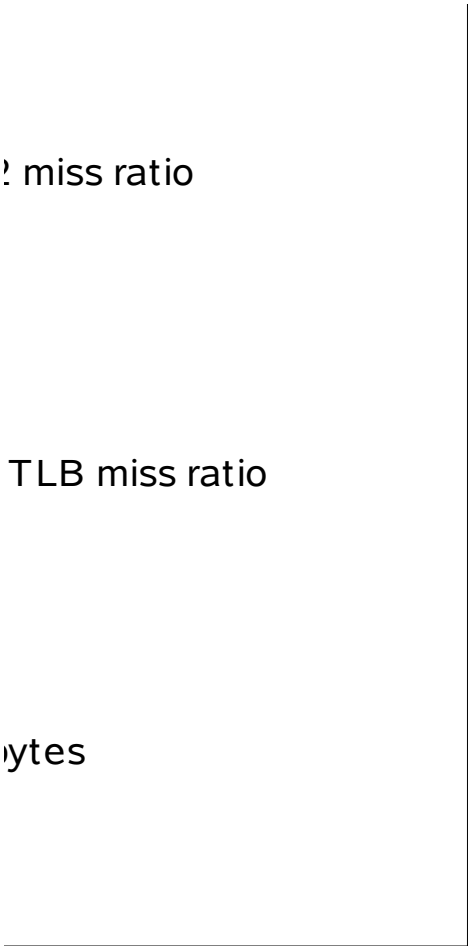


Counters analysis

Scalable Software Services for Life Science

	activesteppot	prepintlist	Cluster 3	Cluster 4	calcepot	Cluster 6
IPC	0,67	0,04	0,68	0,34	0,74	0,41
FXU Mix	0,53	18,63	0,45	1,17	0,34	0,40
FPU Mix	0,12	0,00	0,17	0,11	0,12	0,15
FMA %	0,23	0,00	0,49	0,01	0,21	0,26
Long FP mix	0,00	0,00	0,00	0,00	0,01	0,01
Sore Mix	0,19	0,02	0,13	0,22	#VALOR!	0,23
Load Mix	0,70	18,66	0,49	1,34	0,56	0,61
Mem bytes	0,73	371,32	0,92	1,82	#VALOR!	1,60
TLB miss ratio	0,22	106,45	0,85	0,74	#VALOR!	2,32
L2 miss ratio	1,06	47,55	0,96	3,12	0,15	2,91
L1 miss ratio	36,74	136,12	32,08	30,68	#VALOR!	7,70

Clusters comparison



DISCRETE conclusions

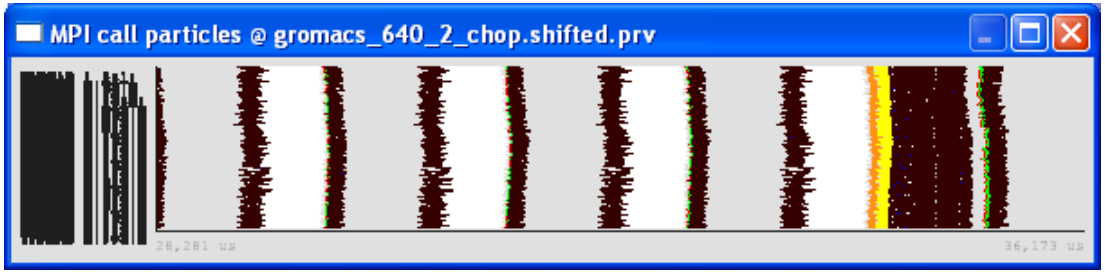
- 5 versions of the code analysed
  - Main improvement from reducing total instructions
  - Some improvements on memory access
- IPC has been improved for half of the main relevant regions but still place for improvement…

GROMACS @ hopper

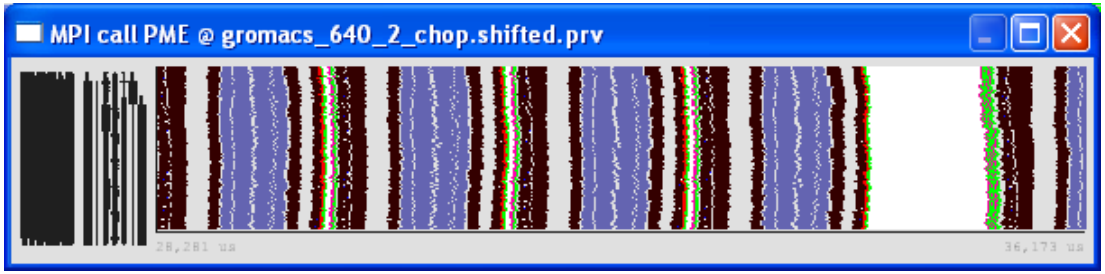
Code structure

- Two kind of tasks: PMEs, particles
- 512 particle tasks, relevant input case?
- New version parallelise PMEs (OpenMP)

particles



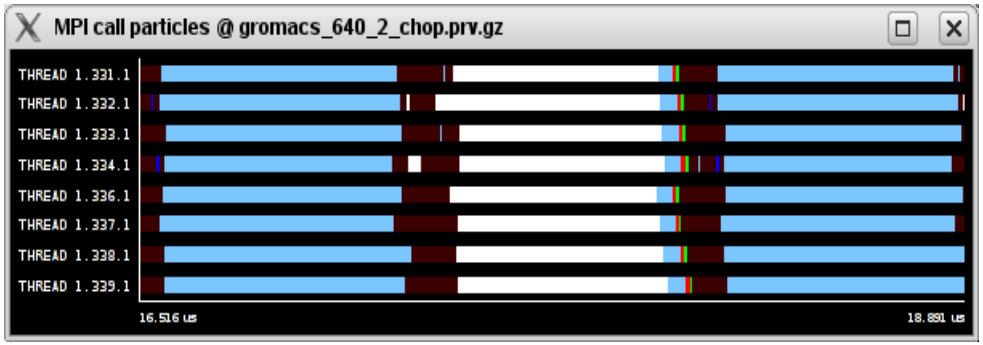
PMEs

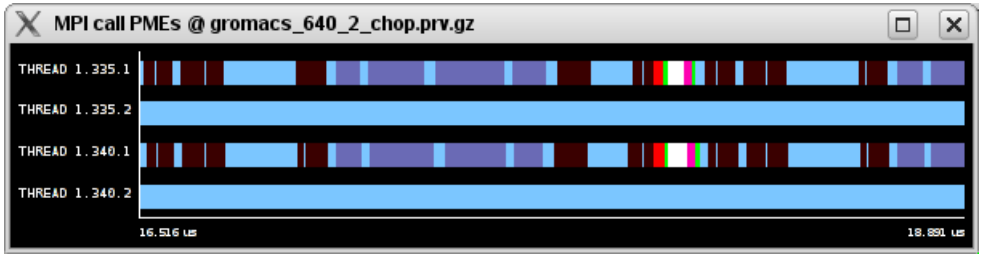


You can ignore small deviations between tasks - due to different clocks

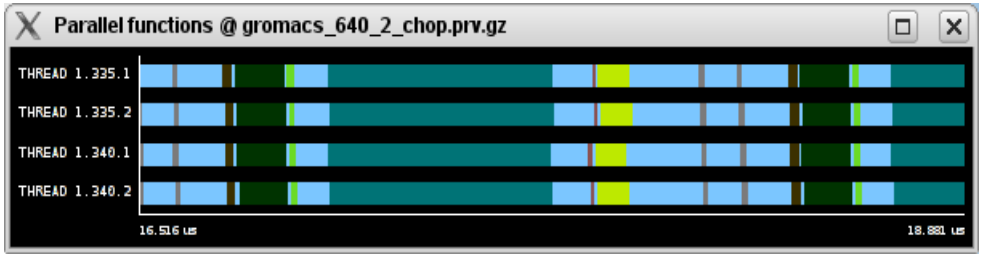
Code structure

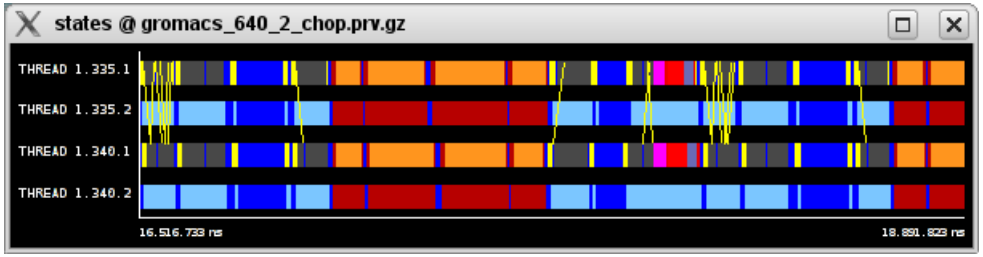
- Why PMEs?





- How?

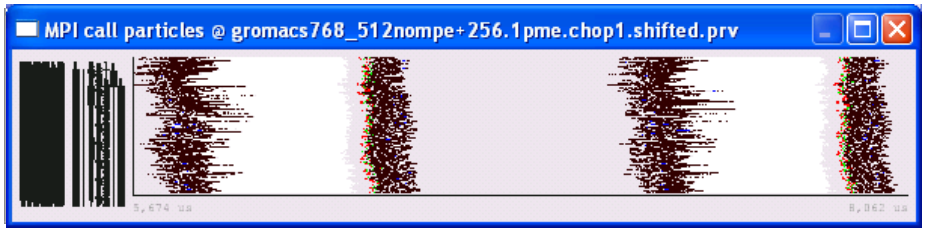


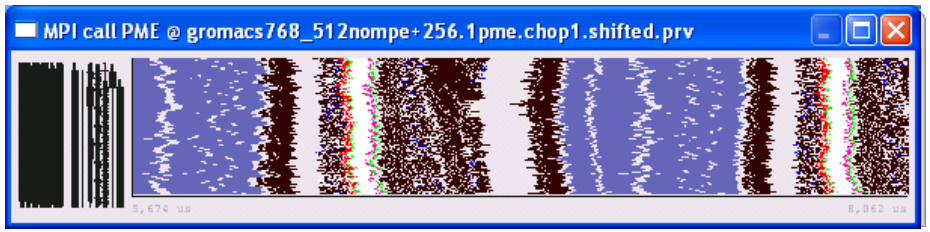


Comparing runs

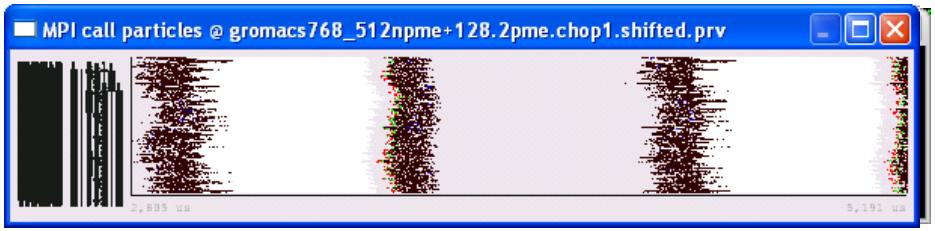
- 512 particle tasks – same resources

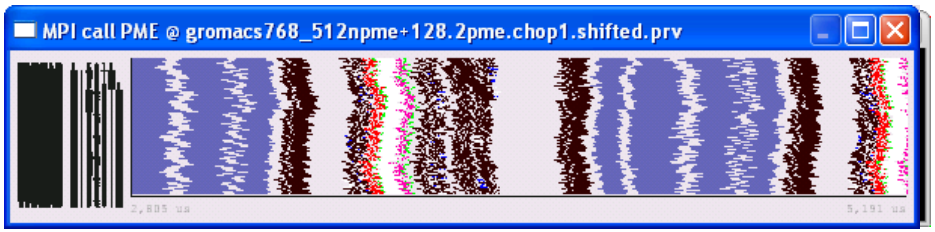
256 PME tasks (768 CPUs)





128 PME tasks x 2 thread (768 CPUs)





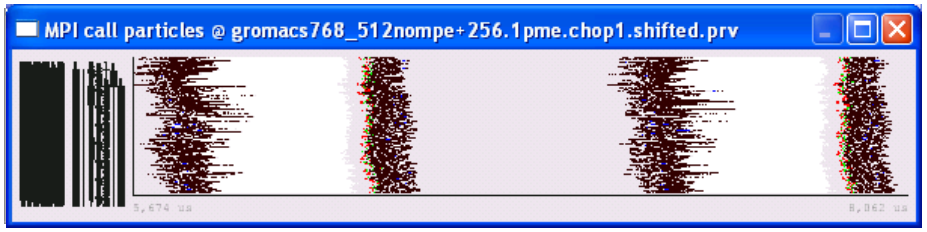
Same amount of resources
"*Same*" ratio between nonPMEs/PMEs w.r.t. computation (2:1) vs (4:2)
Different ratio between nonPMEs/PMEs w.r.t. communication (2:1) vs (4:1)
Both populate a node (24cpus) with 16 nonPME + 8 PME

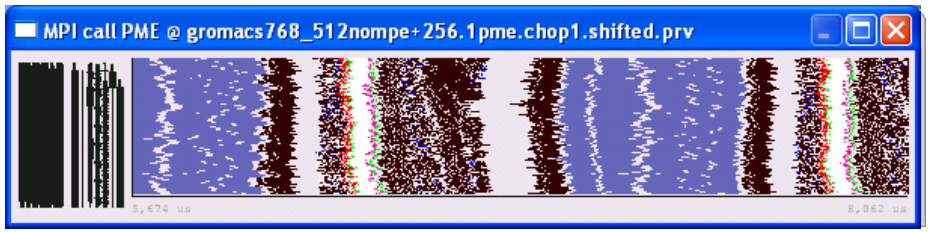
A little bit poor performance with OpenMP – seems like non busy waiting!

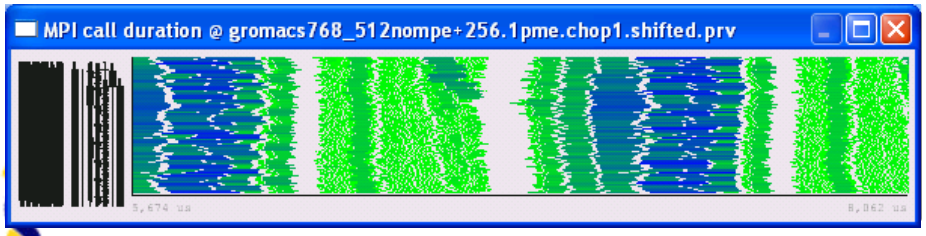
Comparing runs

- 512 particle tasks – same resources

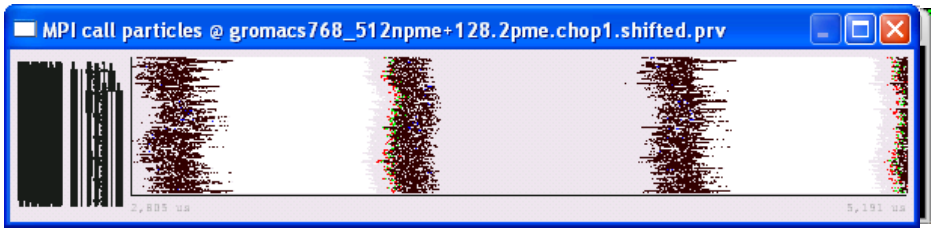
256 PME tasks (768 CPUs)

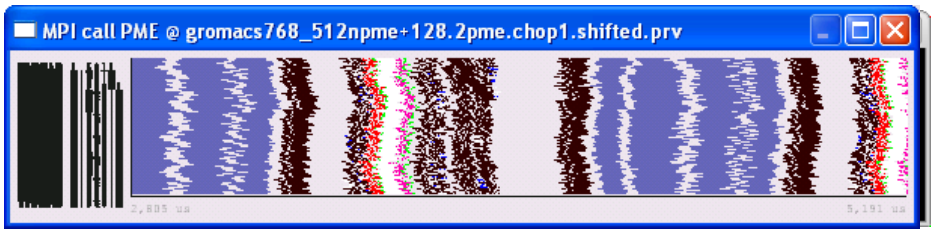


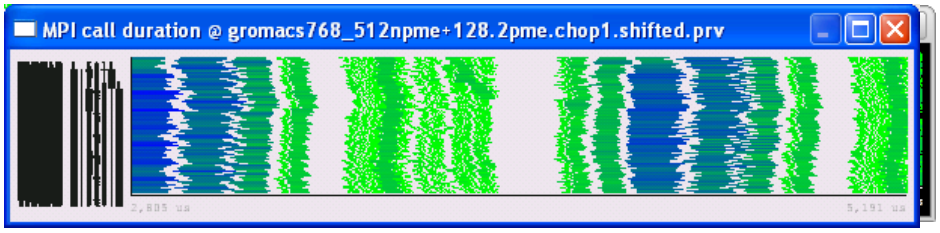




128 PME tasks x 2 thread (768 CPUs)





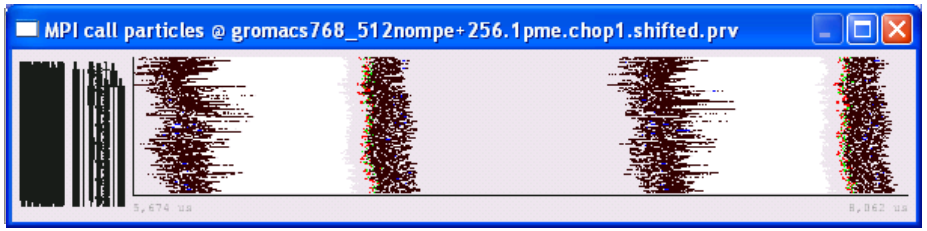


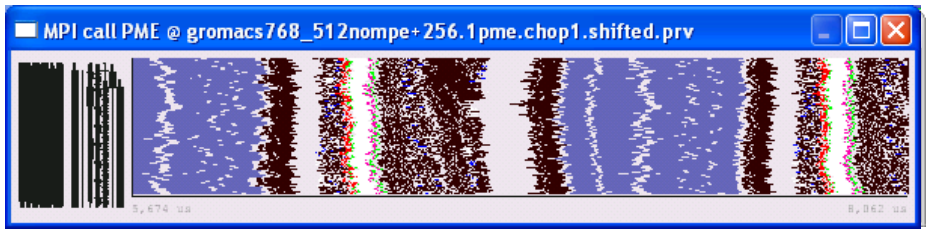
Perturbation on the PMEs All2all seems reduced
Seems correlated with the duration of particles point 2point phase

Comparing runs

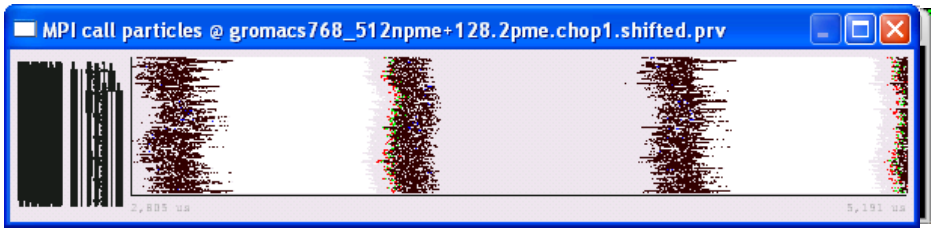
- 512 particle tasks – same resources

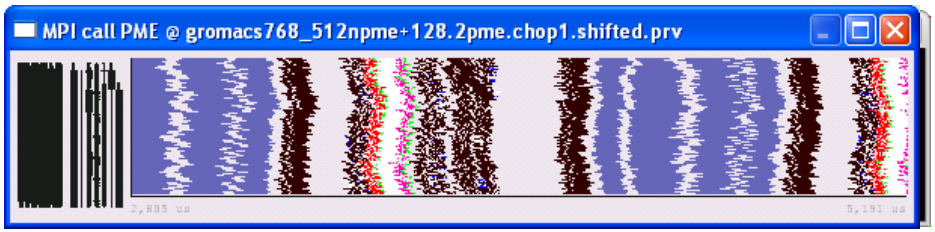
256 PME tasks (768 CPUs)



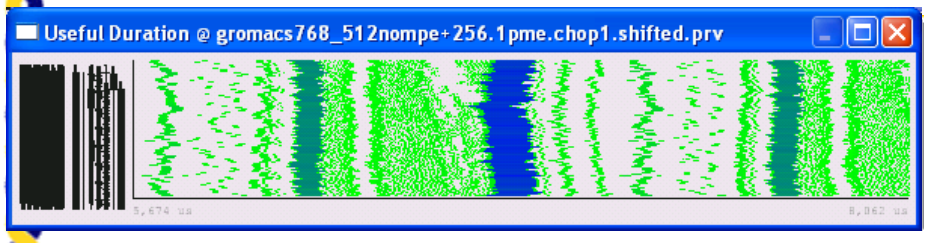


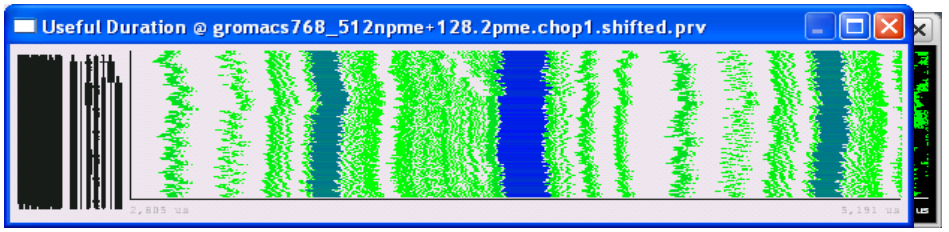
128 PME tasks x 2 thread (768 CPUs)





PMEs computation similar but a little bit worst with OpenMP

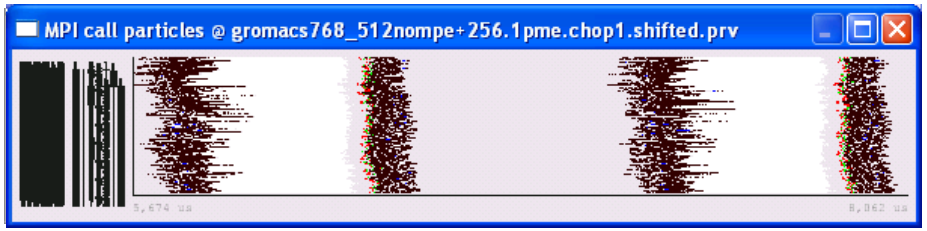


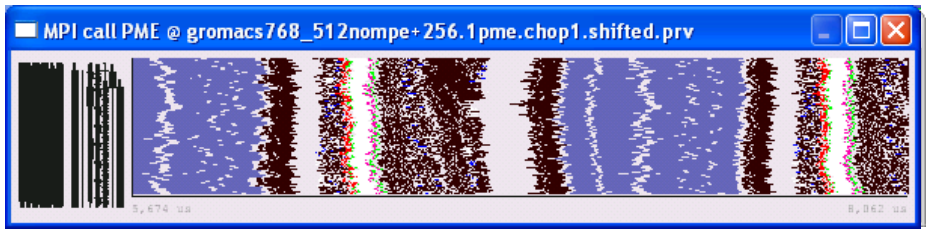


Comparing runs

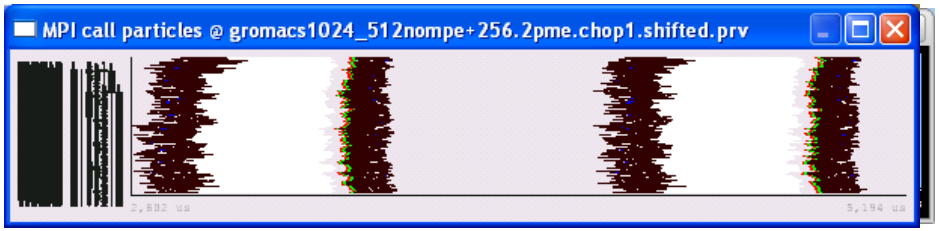
- 512 particle tasks – more resources

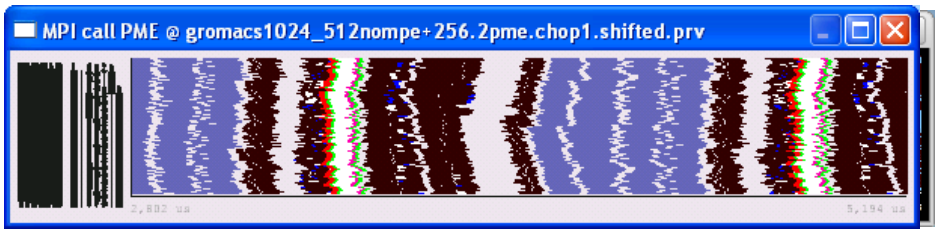
256 PME tasks (768 CPUs)





256 PME tasks x 2 threads (1024 CPUs)





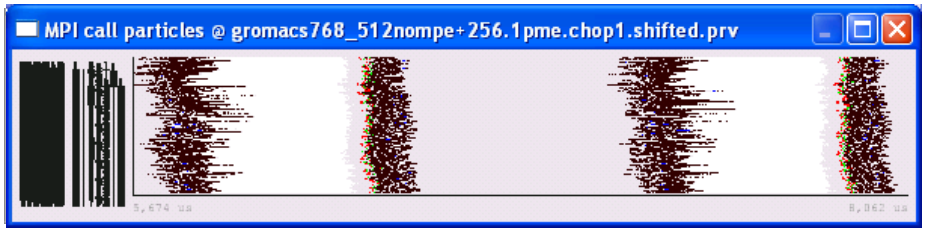
More resources on the right (768 vs 1024)
Different ratio between nonPMEs/PMEs w.r.t. computation (2:1) vs (2:2)
Same ratio between nonPMEs/PMEs w.r.t. communication (2:1) vs (2:1)
Node population: 16 nonPME + 8 PME vs 12 nonPME + 12 PME

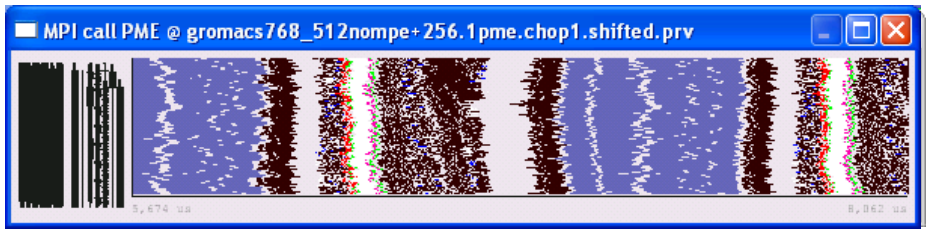
A little bit better performance with OpenMP but 33,3% more resources!

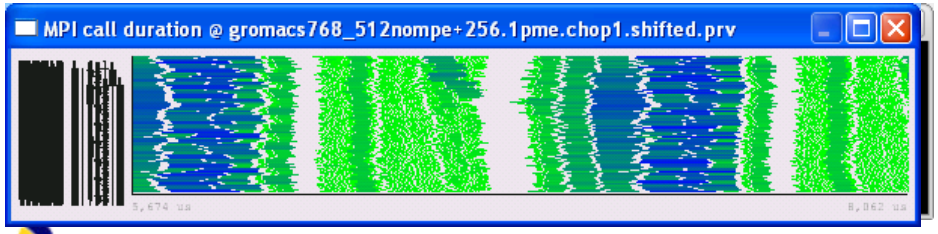
Comparing runs

- 512 particle tasks – more resources

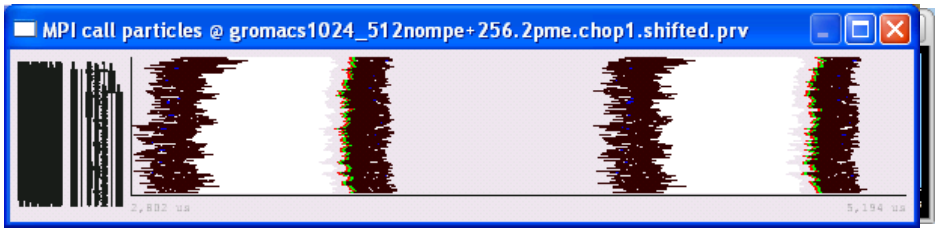
256 PME tasks (768 CPUs)

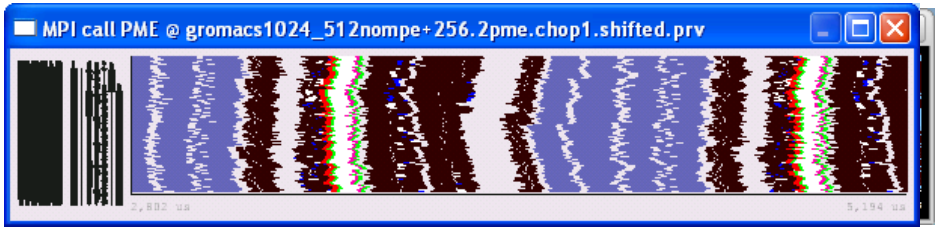


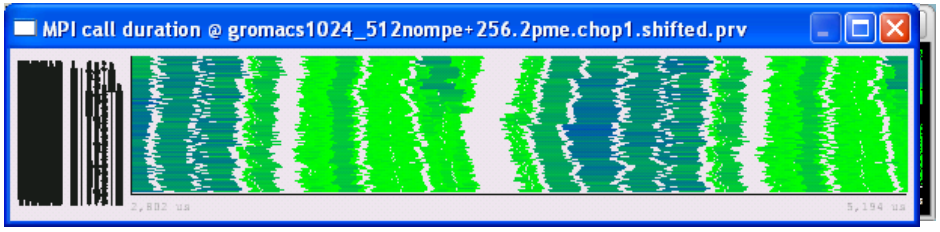




256 PME tasks x 2 threads (1024 CPUs)





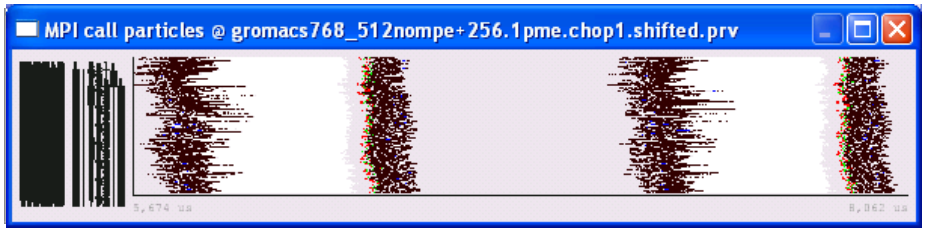


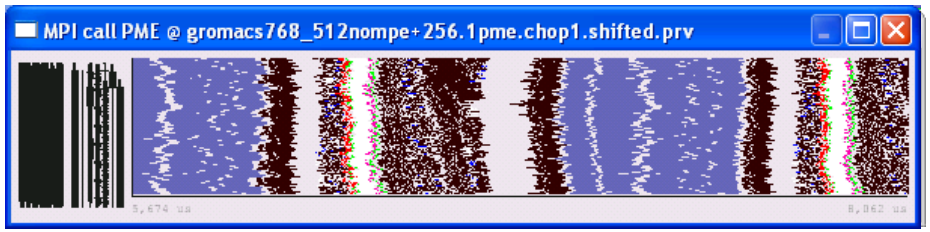
Perturbation on the PMEs All2all significantly reduced
The duration of particles point 2point phase some reduction

Comparing runs

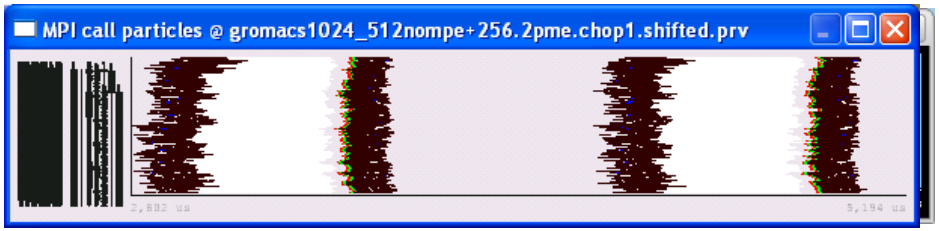
- 512 particle tasks – more resources

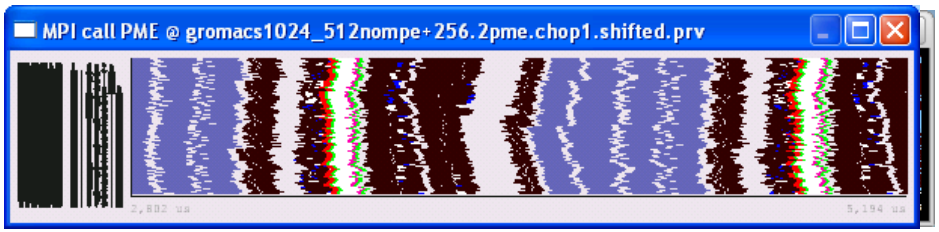
256 PME tasks (768 CPUs)



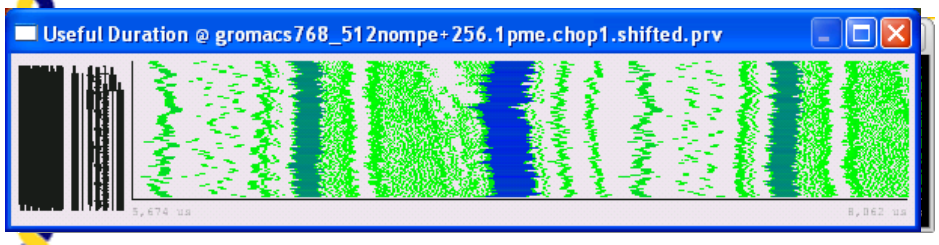


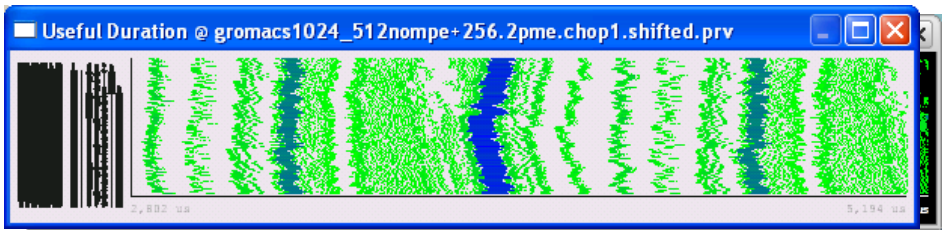
256 PME tasks x 2 threads (1024 CPUs)





PMEs computation improved because uses more resources
OpenMP overhead (hole bigger than computation)

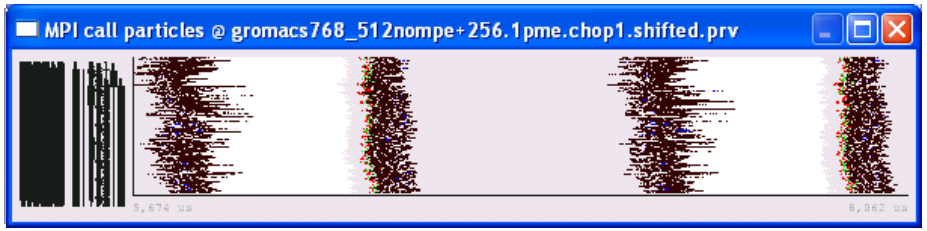


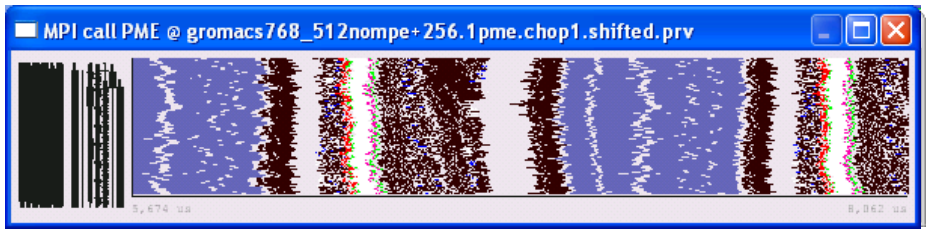


Comparing runs

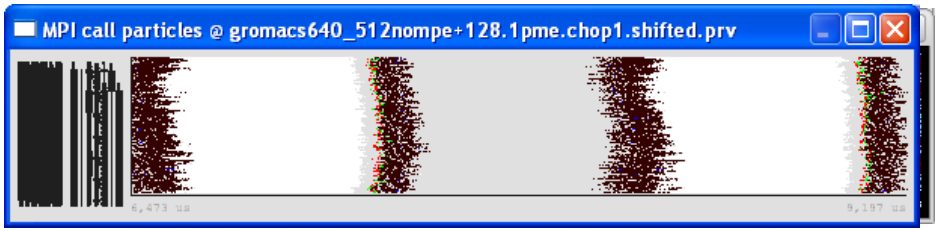
- 512 particle tasks – less resources, no OMP

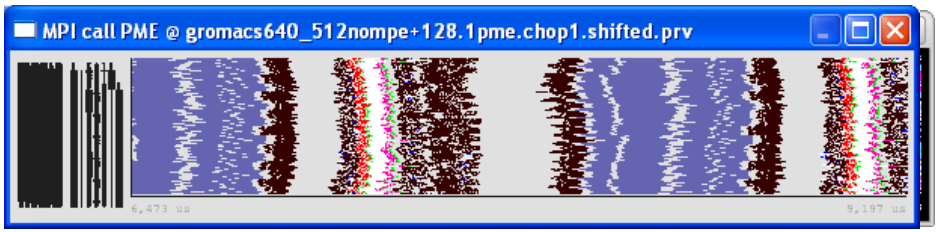
256 PME tasks (768 CPUs)





128 PME tasks(640 CPUs)





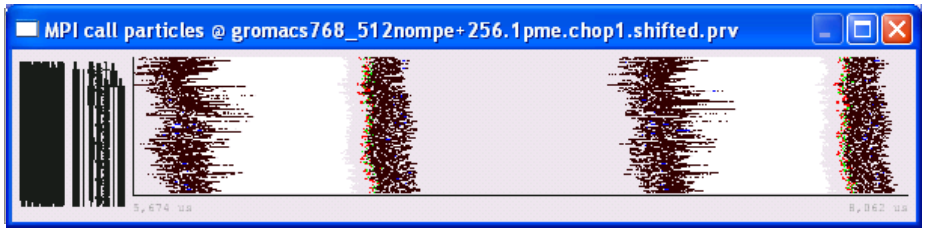
Less resources on the right (768 vs 640)
Different ratio between nonPMEs/PMEs w.r.t. computation (2:1) vs (4:1)
Different ratio between nonPMEs/PMEs w.r.t. communication (2:1) vs (4:1)
No OpenMP
Node population: 16 nonPME + 8 PME vs (variable!!) 20 nonPME + 4 PME

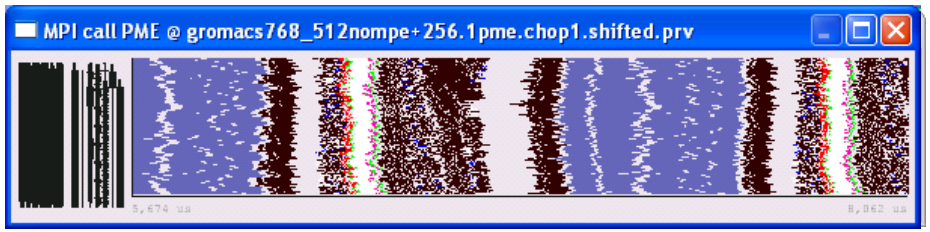
A little bit poor performance with 16.6% less resources

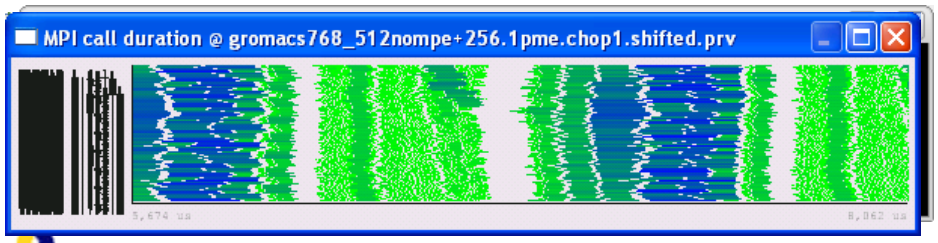
Comparing runs

- 512 particle tasks – less resources, no OMP

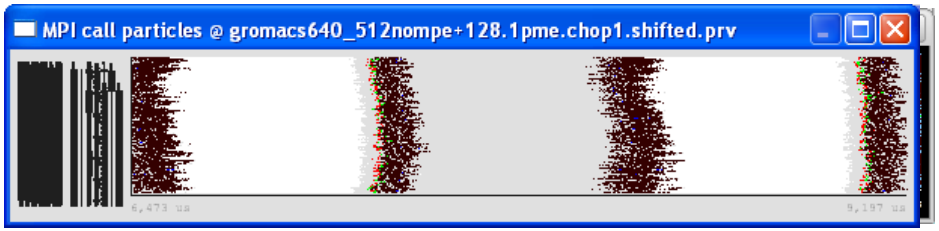
256 PME tasks (768 CPUs)

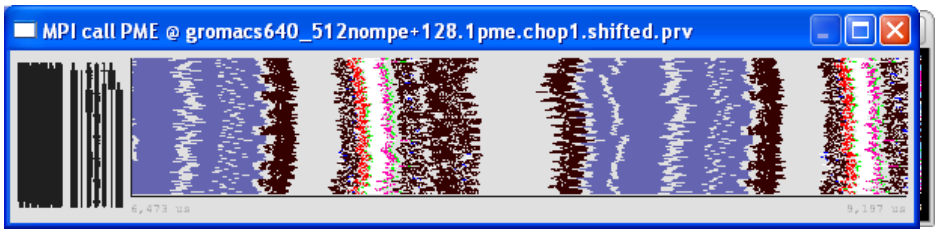


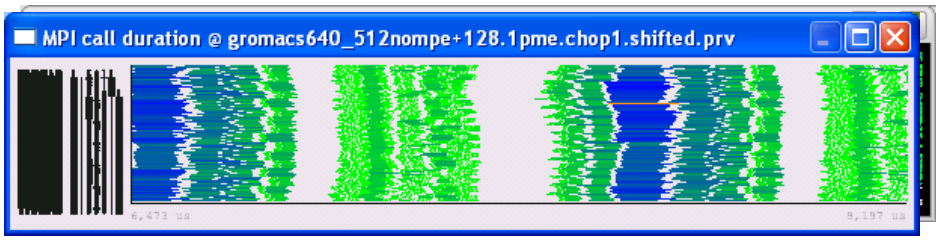




128 PME tasks(640 CPUs)





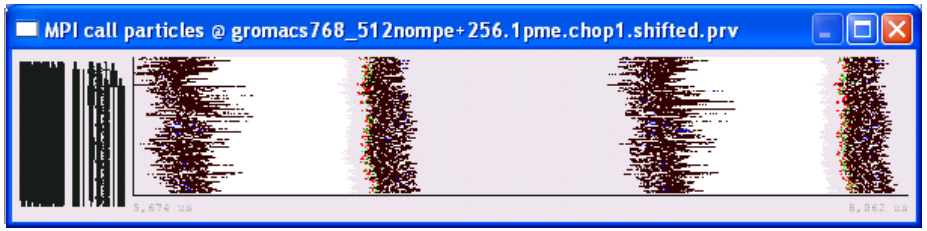


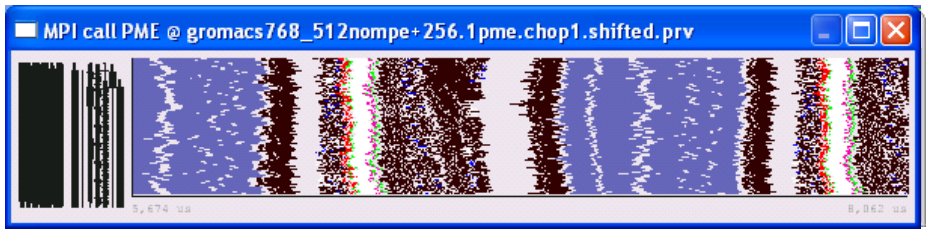
Perturbation on the PMEs All2all more concentrated
The duration of particles point 2point phase may have some reduction

Comparing runs

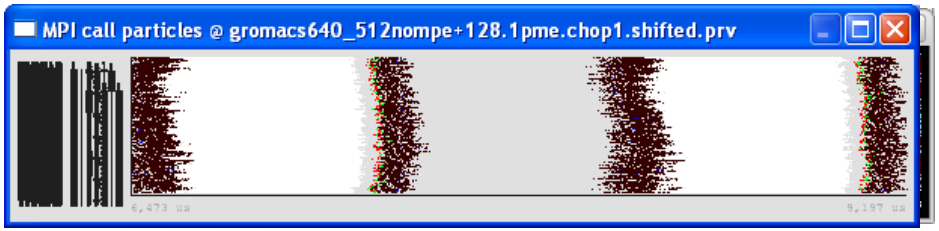
- 512 particle tasks – less resources, no OMP

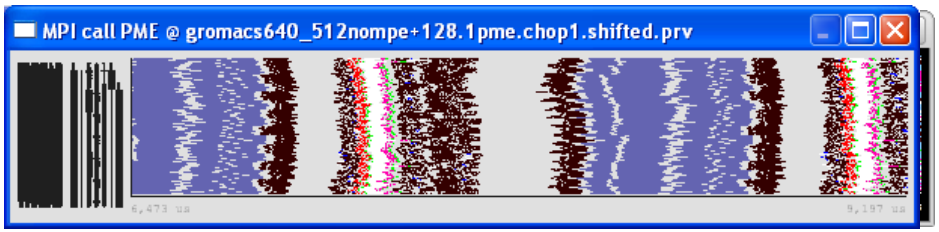
256 PME tasks (768 CPUs)



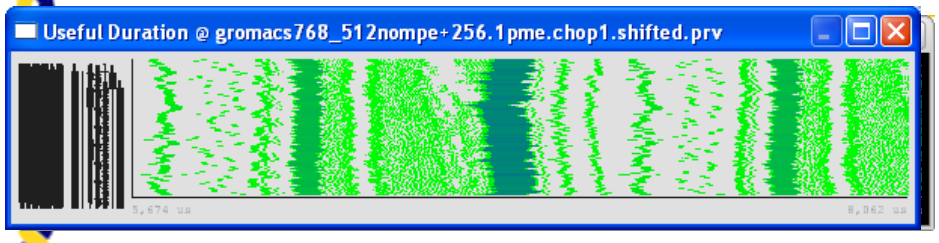


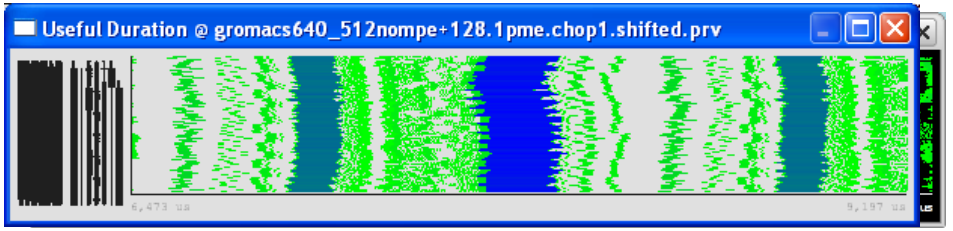
128 PME tasks(640 CPUs)





PMEs computation takes more time because uses half PMEs

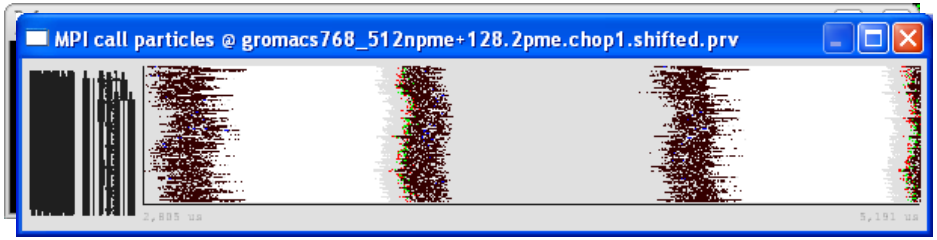


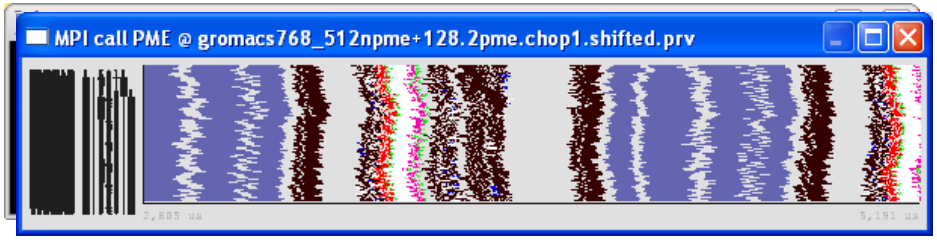


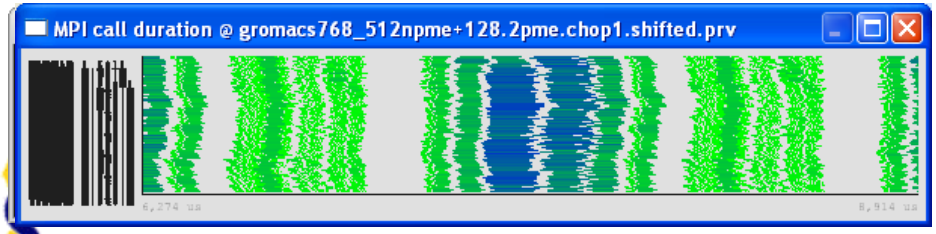
Comparing runs

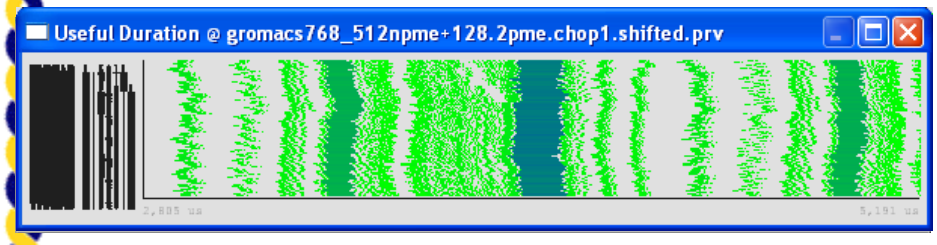
- 512 particle tasks – last comparison

128 PME tasks x 2 thread (768 CPUs)

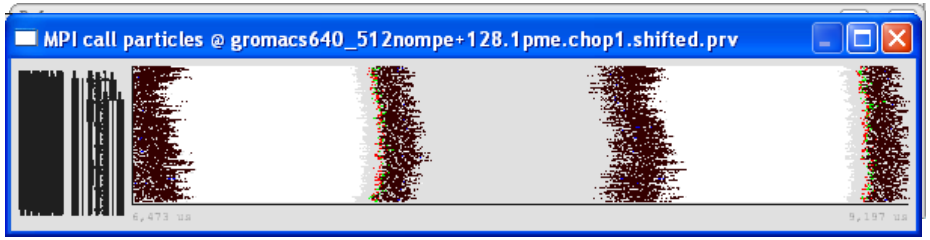


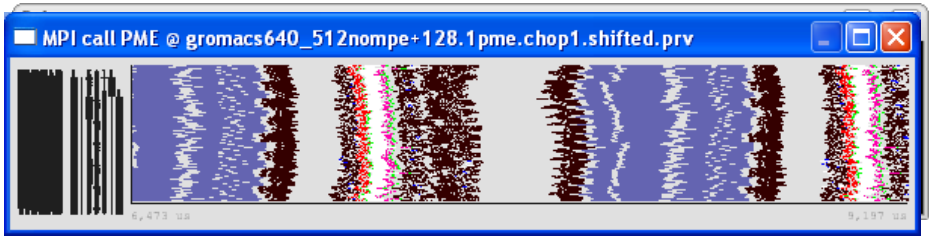


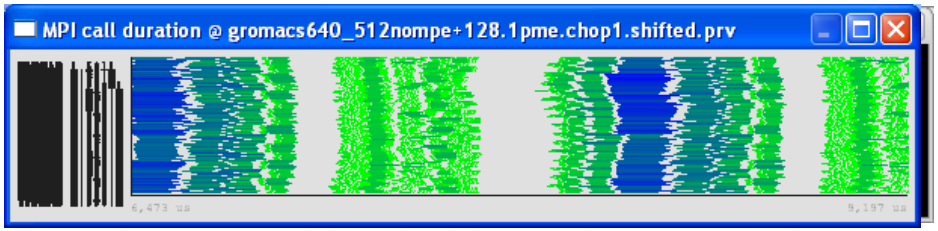


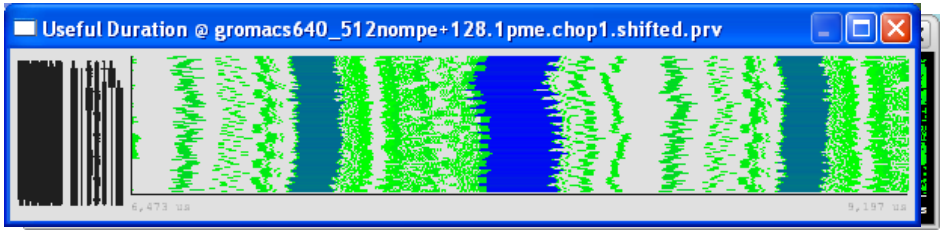


128 PME tasks(640 CPUs)









GROMACS conclusions

- There is not a clear best configuration
  - huge number of pieces!
  - OpenMP overhead looks high (and the OpenMP granularity is fine grain). Seems like threads are non busy waiting
- MPI is the real bottleneck
  - Have tried to map tasks fulling nodes with PMEs/nonPMEs instead mixing them?
  - May be good to try adding barriers to avoid at the same time some processes with point2point while others are on collectives
  - Can the PMEs reduce the number of All2all?