INTELLECTUAL PROPERTY RIGHTS NOTICE:

- The User may only download, make and retain a **copy of the materials for his/her use for non-commercial** and research purposes.
- The User **may not commercially use the material**, unless has been granted prior written consent by the Licensor to do so; and **cannot remove, obscure or modify copyright notices**, text acknowledging or other means of identification or disclaimers as they appear.
- For further details, please contact **BSC-CNS** patc@bsc.es

PATC training, Barcelona, May 2012 ‹#›

PRACE TRAINING COURSE under PRACE Advance Training Centre at BSC

BSC-CNS http://www.bsc.es/

PRACE project http://www.prace-ri.eu/

PRACE Training Portal http://www.training.prace-ri.eu/

PATC @ BSC Training Program

http://www.bsc.es/marenostrum-support-services/hpc-trainings/prace-trainings

PATC training, Barcelona, May 2012 ‹#›

Performance analysis with BSC-Tools

Judit Gimenez (judit@bsc.es)
BSC

Performance analysis tools objective

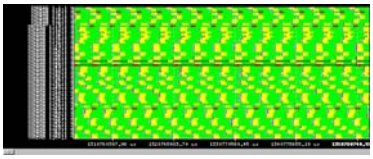
Help generate hypotheses

Help validate hypotheses

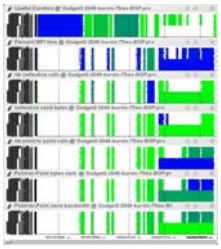
Qualitatively

Quantitatively

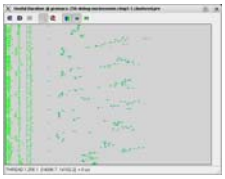
Our Tools



Since 1991
Based on traces
Open Source
http://www.bsc.es/paraver



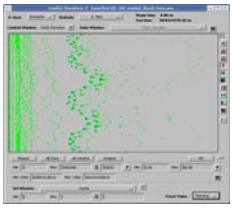
Core tools:
Paraver (paramedir) – offline trace analysis
Dimemas – message passing simulator
Extrae – instrumentation



Focus
Detail, flexibility, intelligence

BSC tools @ PATC, Barcelona May 21-22 2012 5

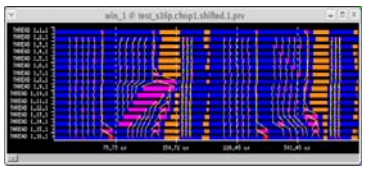
Why traces?



Detail and variability are important
along time, across processors

Highly non-linear systems
microscopic effects may have macroscopic impact

Extremely useful to develop/test analysis techniques



BSC tools @ PATC, Barcelona May 21-22 2012 6

Course outline

Extrae
Paraver
  General description
  Examples
Demos + hands-on

Advanced Analysis
  Scaling model
  Structure detection
  Folding: Instrumentation + sampling
Dimemas
Demos + hands-on

Extrae

Extrae – traced information

Parallel programming model runtime
- MPI, OpenMP, Pthreads, StarSs, ...

Counters
- CPU counters – PAPI (standard + native)
- Network counters (GM, MX) – at flushes
- OS counters

Links to source
- Callstack at MPI calls
- User functions selected – default none

Periodic samples
- PAPI counters + callstack

User events

Extrae – two phases

Run
- Adapt job launching script
- Adapt .xml that controls information to be captured
- Run to generate one .mpit file per process

Merge
- In parallel, typically using less processors than run

Combined Run/Merge
- The run control .xml can configure extrae to also perform the merge

Typical production job script

job.sh

```
#!/bin/bash
…
# @ total_tasks = 2
# @ cpus_per_task = 1
# @ tasks_per_node = 2
…
srun            ./my_MPI_binary
```


Instrumenting an application

job.sh

```
#!/bin/bash
…
# @ total_tasks = 2
# @ cpus_per_task = 1
# @ tasks_per_node = 2
…
srun ./trace.sh ./my_MPI_binary
```

Reference trace.sh and extrae.xml provided with distribution

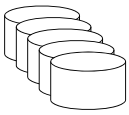
trace.sh

```
export#!/bin/sh

export EXTRAE_HOME=/gpfs/apps/CEPBATOOLS/extrae
export EXTRAE_CONFIG_FILE=extrae.xml

export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitrace.so

## Run the desired program
$*
```



TRACExxxxxx.mpit

TRACE.mpits

TRACE.sym

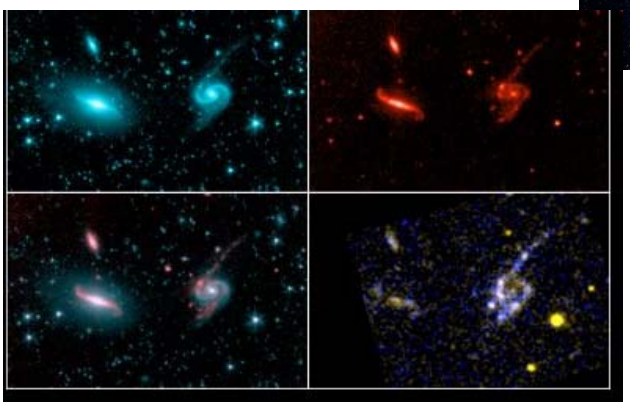
Paraver

Multispectral imaging

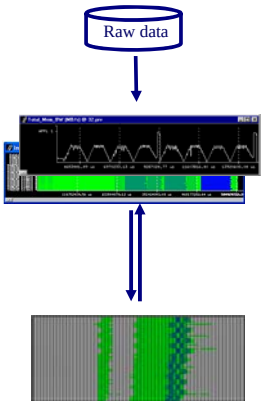
Different looks at one reality

- Different spectral bands (light sources and filters)

Highlight different aspects



Paraver – Performance data browser



Goal = Flexibility
- No semantics
- Programmable

Configuration files
- Distribution
- Your own

Comparative analyses
- Multiple traces
- Synchronize scales

Views: Timelines

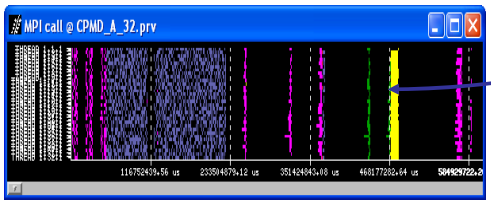
Semantic:

Raw events → Piece-wise constant functions of time for every object

Display:

function values → rendered color

Objects
- Process dimension
  - **Thread**
  - Process
  - Application
  - Workload
- Resource dimension
  - Core
  - Node
  - System



Color encoded value (semantic value)
- Categorical
- Gradient
- Extremes

Time

Timelines

Each window displays one view

Piecewise constant function of time $S(t) = S_i, i \in [t_i, t_{i+1})$

One such function of time per object:

Thread, process, application, workload, CPU, node

Types of functions

Categorical $S_i \in [0, n] \subset N$

State, user function, outlined routine

Logical $S_i \in \{0,1\}$

In specific user function, In MPI call, In long MPI call

Numerical $S_i \in R$

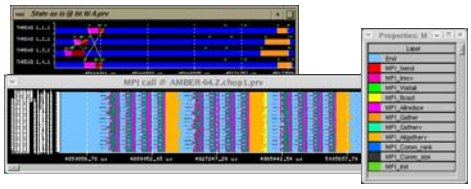
IPC, L2 miss ratio, Duration of MPI call, duration of computation burst

Views: Timelines

Representation

Function of time

Color encoding



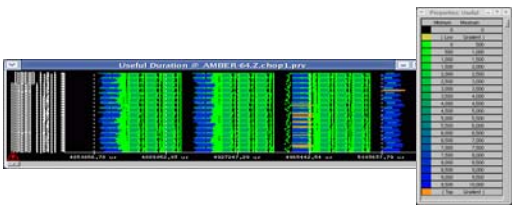
Gradient color

Light green → Dark blue

Not null gradient

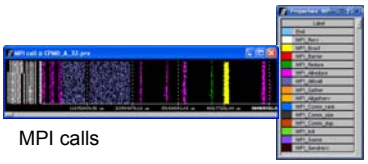
Black for zero value

Light green → Dark blue

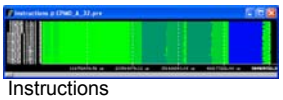


Views: Timelines

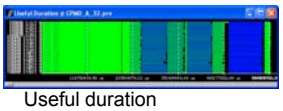
- Basic metrics



MPI calls



Instructions

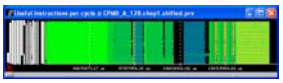


Useful duration

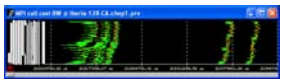
Color encoding

- Derived metrics

$$useful_IPC = \frac{\#instr}{\#cycles} * useful$$

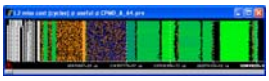


$$MPI_call_Cost = \frac{MPI_call_duration}{\#bytes}$$



- Models

$$L2_{miss_{latency}} = \frac{cycles - instr/idealIPC}{L2misses}$$

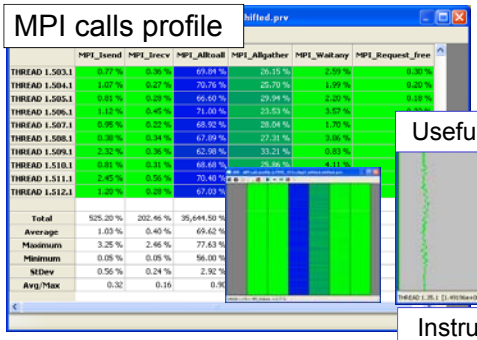


$$preempted_{time} = elapsed - \frac{cycles}{clock_{freq}}$$

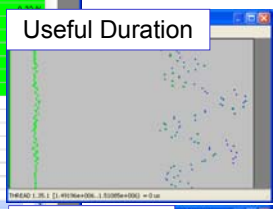
Views: Tables (Profiles, histograms, correlations)

Huge number of statistics computed from timelines

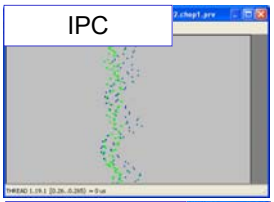
MPI calls profile



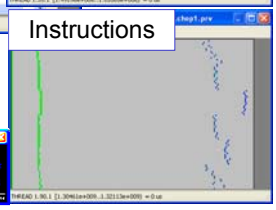
Useful Duration



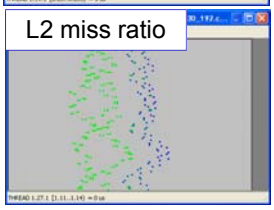
IPC

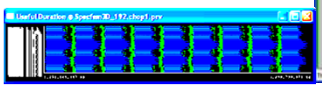


Instructions



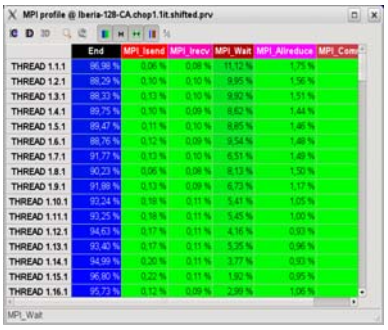
L2 miss ratio

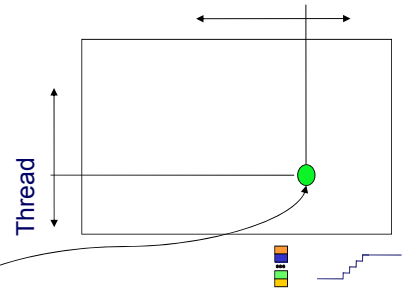




How to read profiles

One columns per specific value of categorical **Control window**



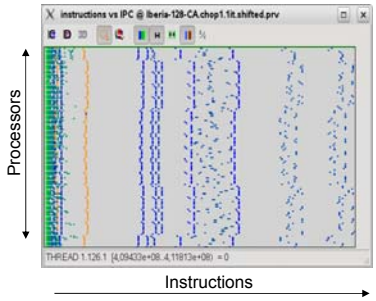


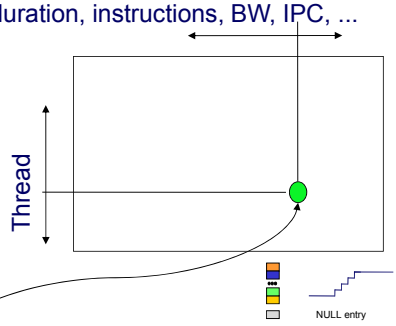
Value/color is a statistic computed for the specific thread
when control window had the value corresponding to the column

Relevant statistics:
Time, %time, #bursts, Avg. burst time
Average of **Data window**

How to read histograms

Columns correspond to bins of values of a numeric **Control window**





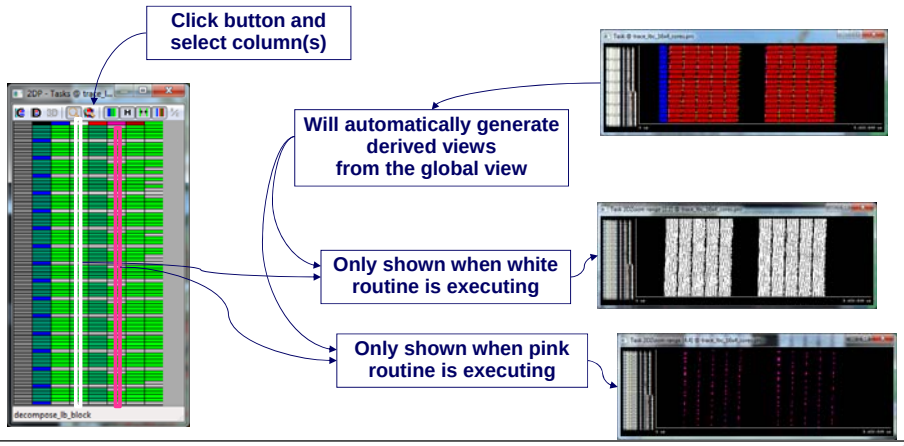
Value/color is a statistic computed for the specific thread
when control window had the value corresponding to the column

Relevant statistics:
Time, %time, #bursts, Avg. burst time
Average of **Data window**

From tables to timelines

Where in the timeline do the values in certain table columns appear?

ie. want to see the time distribution of a given routine?

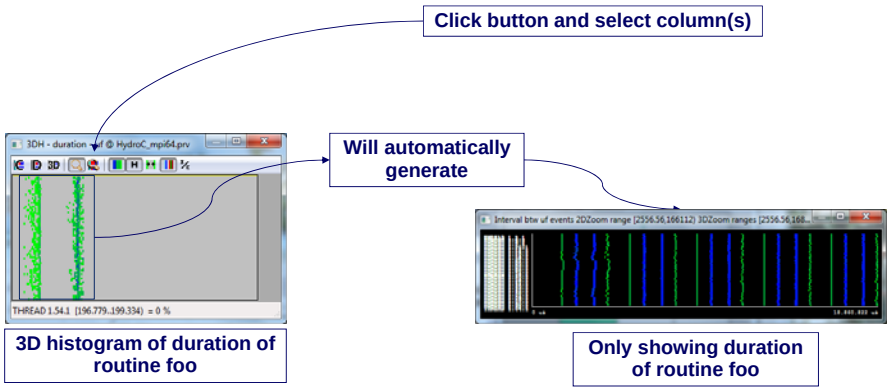


BSC tools @ PATC, Barcelona May 21-22 2012 23

From tables to timelines

Where in the timeline do the values in certain table columns appear?

ie. want to see where the timeline happen computation bursts of a given length?



BSC tools @ PATC, Barcelona May 21-22 2012 24

Comparative analyses

Possible to load several traces

Copy and paste

right click menu

From one window to another: time, duration, size, objects displayed, …

Time between windows and tables: analysis will be computed only for the selected time interval

Distribution cfg directories

General: including a basic views and analysis directories as well as a directory with views on the user functions and call stack.

Counters_PAPI: Hardware counter derived metrics. A generic PAPI directory. Grouped in directories for:

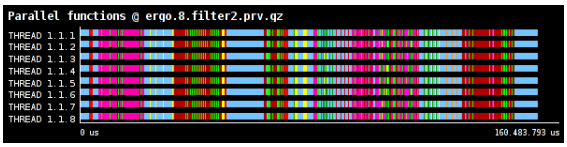
- Program: metrics related to algorithmic/compilation (ie. 3instructions, FP ops,…)
- Architecture: metrics related to execution on specific architecture but not time (i.e. cache misses,….)
- Performance: metrics reporting rates per time unit (i.e. MFLOPS, MIPS, IPC,…)

MPI: Grouped in directories displaying views (timelines) analysis (2/3D profiles and histograms). And further separated into point to point and collectives.

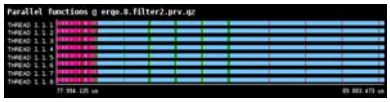
OpenMP: Grouped in directories displaying views (timelines) analysis (2/3D profiles and histograms).

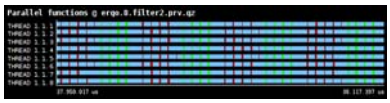
Example 1: Analysing ERGO OpenMP version

ERGO – 8 OpenMP threads : Parallel functions view

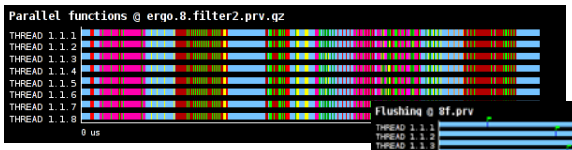


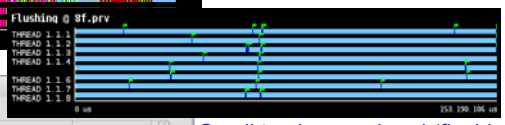
- Structure? Very different phases
- Some phases dominated by sequential (light blue) and would not scale





Example 1: Analysing ERGO OpenMP version



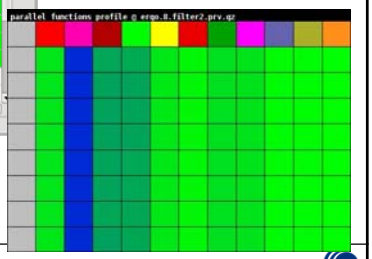


parallel functions profile @ ergo.8.filter2.prv.gz

	THREAD 1.1.1	THREAD 1.1.2	THREAD 1.1.3
compute_operator_matrix_sparseBasisInfoStruct_par_region0	1,48 %	1,48 %	1,48 %
mat6VectorMatrix__par_region0	12,51 %	12,51 %	12,51 %
mat6VectorgemvMatrix_par_loop0_2	5,16 %	5,42 %	5,45 %
mat6VectorgemvMatrix_par_loop1_2	4,92 %	5,27 %	5,28 %
mat6Matrixgemm_par_loop0	1,36 %	1,38 %	1,38 %
mat6Matrixgemm_par_loop1	0,45 %	0,45 %	0,45 %
mat6Matrixsymm_par_loop0	1,84 %	1,84 %	1,82 %
mat6Matrixsysq_par_region0	1,62 %	1,62 %	1,62 %
mat6Matrixsyrk_par_region0	0,18 %	0,18 %	0,18 %
mat6Matrixsyrk_par_region3	0,17 %	0,17 %	0,17 %
mat6Matrixgemm_par_loop2	0,03 %	0,04 %	0,04 %
Total	29,73 %	30,36 %	30,38 %

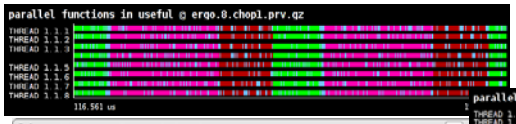
THREAD 1.1.8 mat6Matrixsyrk_par_region3 = 267.971 us

Small tracing overhead (flushing) just the dark blue



- Less than 30% time is parallelised (Amdahls law limits the scalability)
- Loops balanced between threads

Example 1: Analysing ERGO OpenMP version



profile states @ ergo.8.chop1.prv.gz

	Idle	Running	Synchronization	Scheduling and Fork/Join
THREAD 1.1.1	-	91,54 %	2,77 %	5,69 %
THREAD 1.1.2	4,64 %	92,87 %	1,58 %	0,90 %
THREAD 1.1.3	5,48 %	91,18 %	2,54 %	0,79 %
THREAD 1.1.4	5,83 %	90,83 %	2,48 %	0,86 %
THREAD 1.1.5	5,84 %	90,98 %	2,25 %	0,92 %
THREAD 1.1.6	6,22 %	91,17 %	1,68 %	0,94 %
THREAD 1.1.7	6,09 %	90,06 %	2,74 %	1,11 %
THREAD 1.1.8	4,82 %	93,22 %	1,16 %	0,81 %
Total	38,92 %	731,85 %	17,21 %	12,02 %
Average	5,56 %	91,48 %	2,15 %	1,50 %
Maximum	6,22 %	93,22 %	2,77 %	5,69 %
Minimum	4,64 %	90,06 %	1,16 %	0,79 %
StDev	0,57 %	0,99 %	0,56 %	1,59 %
Avg/Max	0,89	0,98	0,78	0,26

- Bad case: threads idle 85% of the time

profile states @ ergo.8.chop2.prv.gz

	Idle	Running	Scheduling and Fork/Join
THREAD 1.1.1	0,04 %	97,84 %	2,12 %
THREAD 1.1.2	85,98 %	13,78 %	0,23 %
THREAD 1.1.3	86,22 %	13,57 %	0,21 %
THREAD 1.1.4	85,78 %	14,00 %	0,22 %
THREAD 1.1.5	86,07 %	13,72 %	0,21 %
THREAD 1.1.6	85,96 %	13,83 %	0,21 %
THREAD 1.1.7	86,28 %	13,46 %	0,26 %
THREAD 1.1.8	85,92 %	13,88 %	0,20 %
Total	602,24 %	194,10 %	3,65 %
Average	75,28 %	24,26 %	0,46 %
Maximum	86,28 %	97,84 %	2,12 %
Minimum	0,04 %	13,46 %	0,20 %
StDev	28,44 %	27,81 %	0,63 %
Avg/Max	0,87	0,25	0,22

Running

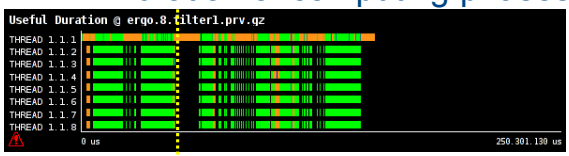
- Good case: 2% on barriers, 1.5% due to dynamic balancing overhead – make big chucks or try guided sched?

Example 1: Analysing ERGO OpenMP version

ERGO – scalability

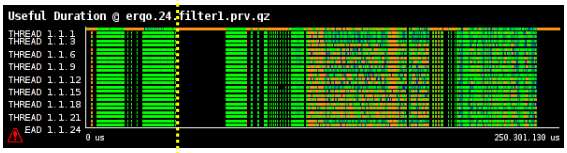
Duration of computing phases

8 OpenMP threads



16 OpenMP threads

24 OpenMP threads



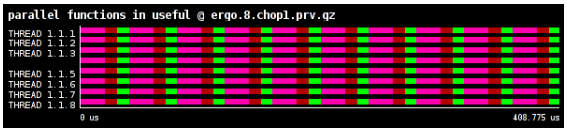
Not reduced with more threads

Even takes longer when scaling #threads

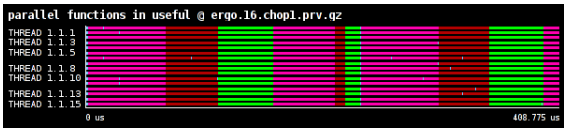
Example 1: Analysing ERGO OpenMP version

Looking at a highly parallelised phase within the second part. Synchronised views of the parallel functions

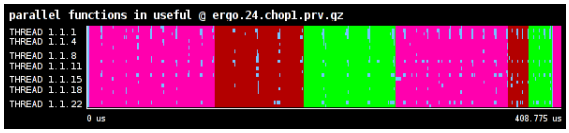
8 OpenMP threads



16 OpenMP threads



24 OpenMP threads

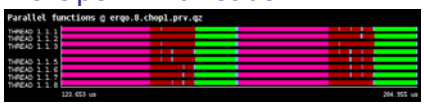


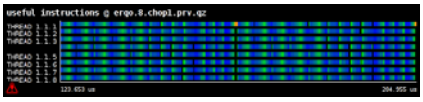
Parallel loops take longer with more threads
With 16 or 24 -> two modes for red and green

Example 1: Analysing ERGO OpenMP version

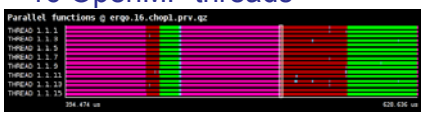
2 iterations, non synchronised views (left: parallel functions, right: instructions while computing)

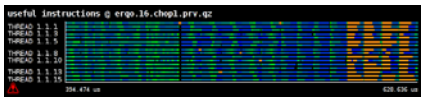
8 OpenMP threads



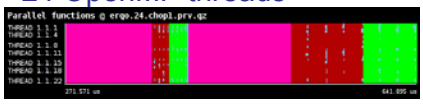


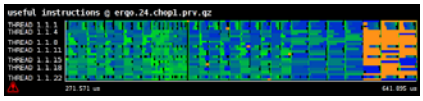
16 OpenMP threads





24 OpenMP threads



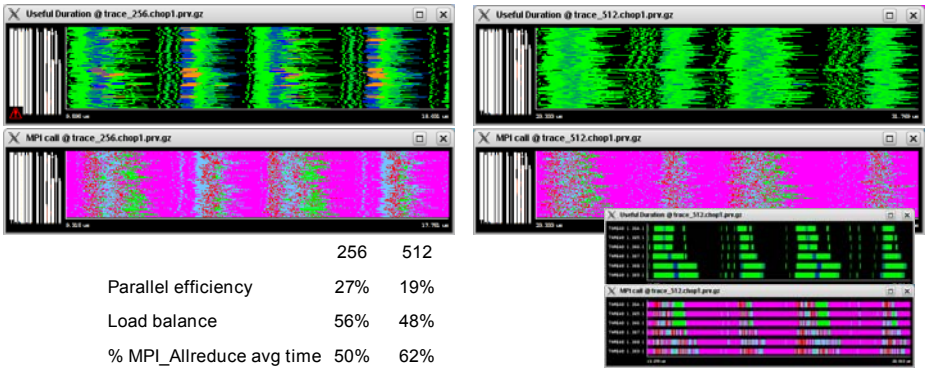


#instructions of green parallel chunks is enlarged

Example 2: Code optimization

Short diagnosis: FrontFlowRed – 256 vs 512 tasks

- Unbalance and MPI time increases
- Sequence of MPI_Allreduce calls
- Poor IPC, high cache misses



	256	512
Parallel efficiency	27%	19%
Load balance	56%	48%
% MPI_Allreduce avg time	50%	62%

Example 2: Code optimization

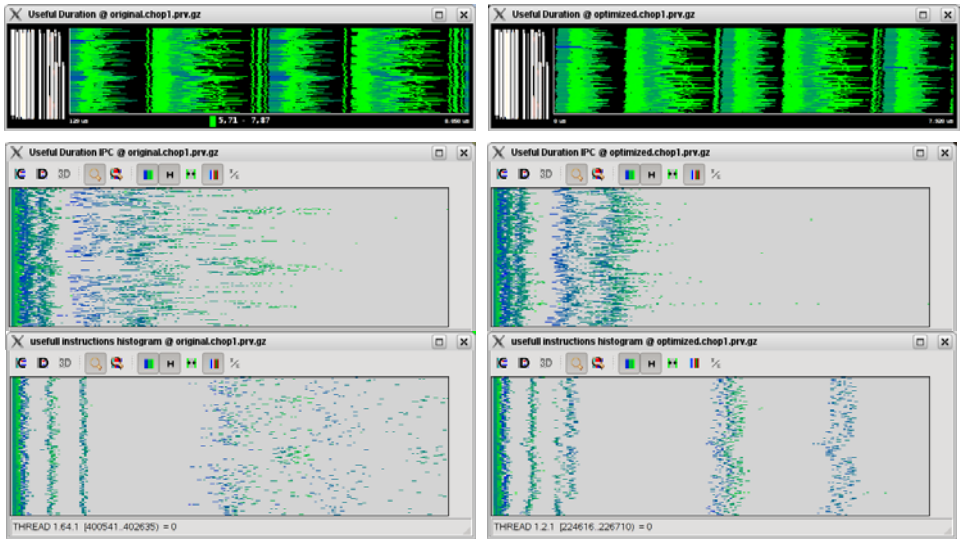
Modifications in the source code

- Improved load balance – weight nodes based on their connectivity, balance total weight
- Reduction of all to all communications – half of MPI_Allreduce calls were eliminated
- Reduce cache misses – reordering node number

Half an hour meeting for the analysis, few days of work to modify code → 25% of improvement

Example 2: Code optimization

256 tasks – Load balance



Example 2: Code optimization

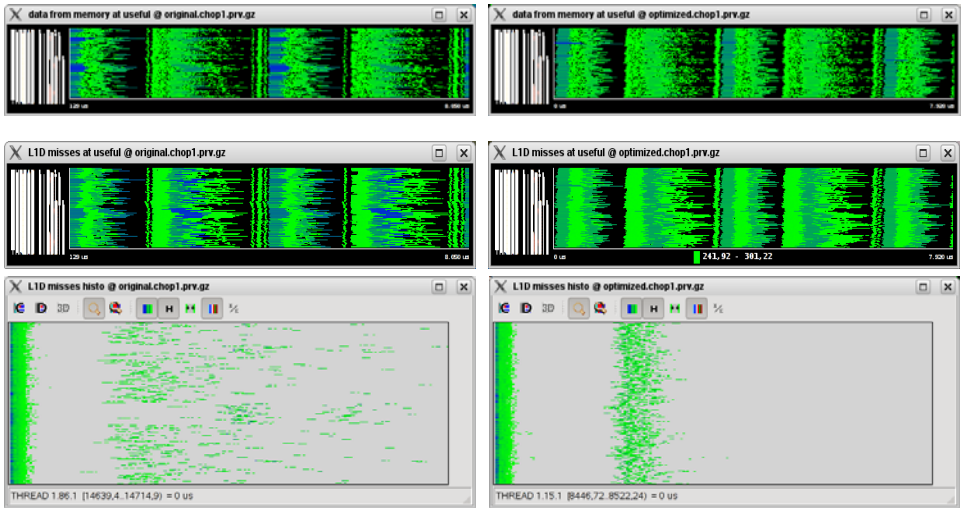
256 tasks – MPI time

	End	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Allreduce
Total	7.617,81 %	2.432,05 %	1.772,21 %	2.245,19 %	11.532,74 %
Average	29,76 %	9,50 %	6,92 %	8,77 %	45,05 %
Maximum	51,85 %	18,11 %	13,00 %	28,59 %	67,69 %
Minimum	17,59 %	1,65 %	1,37 %	1,46 %	14,88 %
StDev	8,04 %	3,33 %	2,39 %	6,05 %	11,03 %
Avg/Max	0,57	0,52	0,53	0,31	0,67

	End	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Allreduce
Total	9.089,05 %	3.100,13 %	2.107,96 %	1.511,02 %	9.791,83 %
Average	35,50 %	12,11 %	8,23 %	5,90 %	38,25 %
Maximum	51,05 %	20,77 %	14,59 %	21,31 %	64,34 %
Minimum	22,99 %	3,47 %	3,21 %	1,72 %	10,90 %
StDev	5,92 %	3,62 %	2,61 %	3,89 %	10,13 %
Avg/Max	0,70	0,58	0,56	0,28	0,59

Example 2: Code optimization

256 tasks – Memory access



Tools web site

www.bsc.es/paraver

- downloads
  - Sources
  - Binaries
- documentation
  - Training guides
  - Tutorial slides

Conclusions

- The complexity of current systems pushes the need for performance analysis tools
- Tried to demonstrate the benefits of looking at the details
- One image is worth 1000 words → learning to "read" them pays off :)
- Do not speculate about your code performance behaviour, look at it

www.bsc.es/paraver

INTELLECTUAL PROPERTY RIGHTS NOTICE:

- The User may only download, make and retain a **copy of the materials for his/her use for non-commercial** and research purposes.
- The User **may not commercially use the material**, unless has been granted prior written consent by the Licensor to do so; and **cannot remove, obscure or modify copyright notices**, text acknowledging or other means of identification or disclaimers as they appear.
- For further details, please contact **BSC-CNS** patc@bsc.es