Understanding applications performance with Paraver

Judit Gimenez
BSC

PATC Parallel Programming Workshop
Oct 14-18 2013
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
Multispectral imaging

Different looks at one reality
- Different spectral bands (light sources and filters)

Highlight different aspects
- Can combine into false colored but highly informative images
Spreadsheets and browsers

- Display, manipulate data
  - Dynamic content
  - User defined operations
A “different” view on performance analysis and tools

**Behavioral structure** vs. **syntactic structure**
- Algorithmic and performance
- In space and time

**Variability**
- Multimodal distributions
- Variability + synchronization → critical non linear effects

**Flexibility** to let analyst navigate the captured data and gain as much **insight** as possible from as **few application runs** as possible.

“That what is simple is rarely understood”

my iPads Shanghai cookies
What is Paraver

- A browser …

- … to manipulate (visualize, filter, cut, combine, …) ….

- … sequences of time-stamped events …

- … with a multispectral philosophy …

- … and a mathematical foundation …

- … that happens to be mainly used for performance analysis
Sequence of time stamped records

- Punctual events
  - Something happened: when and where (object/entity: … thread)
  - One record per specific information (encoded as a type and a value)
    - About the event
    - About the interval from the previous event till this one (i.e. hardware counts,…)

- Relations between objects (… communications)
  - Source and sink
  - Attributes (… size, tag)

- Separate numeric (.prv) and symbolic (.pcf) files

Only information derived from captured events and data can be reported.
- Trivial but … often forgotten
Extrae

Major BSC instrumentation package

When / where
- Parallel programming model runtime
  - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, MIC…
  - API entry/exit, OpenMP outlined routines
- Selected user functions
- Periodic samples
- User events

Additional information
- Counters
  - PAPI
  - Network counters
  - OS counters
- Link to source code
  - Callstack
How does Extrae intercept your app?

**LD_PRELOAD**
- Works on production binaries
- Specific library for each combination of runtimes
- Does not require knowledge on the application

**Dynamic instrumentation**
- Works on production binaries
- Just specify functions to be instrumented.

**Other possibilities**
- Link instrumentation library statically (i.e., PMPI @ BG/Q, …)
- OmpSs (instrumentation calls injected by compiler + linked to library)

<table>
<thead>
<tr>
<th>Programming model</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>libseqtrace</td>
</tr>
<tr>
<td>Pure MPI</td>
<td>libmpitrace[f]^1</td>
</tr>
<tr>
<td>Pure OpenMP</td>
<td>libomptrace</td>
</tr>
<tr>
<td>Pure Pthreads</td>
<td>libpttrace</td>
</tr>
<tr>
<td>CUDA</td>
<td>libcudatrace</td>
</tr>
<tr>
<td>MPI + OpenMP</td>
<td>libomppitrace[f]^1</td>
</tr>
<tr>
<td>MPI + Pthreads</td>
<td>libptmppitrace[f]^1</td>
</tr>
<tr>
<td>Mpi + CUDA</td>
<td>libcudamppitrace[f]^1</td>
</tr>
</tbody>
</table>

^1 for Fortran codes

Based on DynInst
U.Wisconsin/U.Maryland
How to use Extrae?

Adapt job submission script
- Specify LD_PRELOAD library and xml instrumentation control file

Specify the data to be captured in the .xml instrumentation control file

Run and get the trace …

Extrae 2.3.4 User's Guide available in
http://www.bsc.es/computer-sciences/performance-tools/documentation

Default control files and further examples within installation in
$EXTRAE_HOME/share/example
Paraver – Performance data browser

Trace visualization/analysis
+ trace manipulation

Timelines

2/3D tables (Statistics)

Goal = Flexibility
No semantics
Programmable

Comparative analyses
Multiple traces
Synchronize scales
Every behavioral aspect/metric described as a function of time
  - Possibly aggregated along
    • The process model dimension (thread, process, application, workload)
    • The resource model dimension (core, node, system)
  - Language to describe how to compute such functions of time (GUI)
    • Basic operators (from) trace records
    • Ways of combining them

Those functions of time can be rendered into a 2D image
  - Timeline

Statistics can be computed for each possible value or range of values of that function of time
  - Tables: profiles and histograms
Paraver mathematical foundation

\[ s(t) = S_i, t \in [t_i, t_{i+1}), i \in \mathbb{N} \]

Function of time  |  Series of values

Trace

Filter

Semantic

Display

Sub-Trace / subset of records

(S_1, t_1), (S_2, t_2), (S_3, t_3), …
Each window displays one view
- **Piecewise constant** function of time

Types of functions
- Categorical
  - State, user function, outlined routine
- Logical
  - In specific user function, In MPI call, In long MPI call
- Numerical
  - IPC, L2 miss ratio, Duration of MPI call, duration of computation burst

\[ s(t) = S_i, i \in [t_i, t_{i+1}) \]

\[ S_i \in [0, n] \subset N, \quad n < \]

\[ S_i \in \{0, 1\} \]

\[ S_i \in R \]
Representation

- Function of time

- Colour encoding

- Not null gradient
  - Black for zero value
  - Light green → Dark blue

Non linear rendering to address scalability

Basic functions of time

Semantic module

- From Events to functions of time
  - Last event value
  - Next event value
  - Average Next Event Value
  - Interval btw. Events

Composition

- S'(t) = f(S(t))
  - Sign
  - 1-sign
  - Select range
  - Sign * Is equal
  - Delta
  - Stacked value

See slides at end of presentation for details
Tables: Profiles, histograms, correlations

From timelines to tables

**MPI calls**

**Useful Duration**

**Histogram Useful Duration**

**MPI calls profile**
Analyzing variability through histograms and timelines

Useful Duration

IPC

Instructions

L2 miss ratio
Analyzing variability through histograms and timelines

By the way: six months later ….
3D Tables

An additional control dimension
- One table (plane) per value (or range) of 3D window
- i.e. histogram of duration of each function
Where in the timeline do the values in certain table columns appear?

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

Only showing when a given value happens
Variability … is everywhere

- CESM: 16 processes, 2 simulated days

- Histogram useful computation duration shows high variability

- How is it distributed?

- Dynamic imbalance
  - In space and time
  - Day and night.
  - Season? 😊
Other mechanisms integrated in the GUI

- Trace manipulation
  - Cut
  - Filter

- Performance analytics
  - Clustering
  - Folding
  - Tracking

- Executing external commands and tools
  - BSC Tools
  - Scripts
  - External tools
Data handling/summarization capability

- **Filtering**
  - Subset of records in original trace
  - By duration, type, value,…
  - Filtered trace IS a paraver trace and can be analysed with the same cfgs (as long as needed data kept)

- **Cutting**
  - All records in a given time interval
  - Only some processes

- **Software counters**
  - Summarized values computed from those in the original trace emitted as new even types
  - #MPI calls, total hardware count,…

See slides at end of presentation for details
Execute external commands …
- Predefined: Dimemas, Stats,…
- User specified binaries or scripts

... specifying arguments
- Trace
- Command specific arguments

Paramedir
- Non graphical version of Paraver
- Reads trace, applies standard cfgs, writes ASCII output (table,…) 

Scripts can use paramedir, Dimemas, clustering … in parametric sweeps, search/optimization loops,…
Executing external commands

Example: basic_analysis.py

Analysis of XXXXX.prv

Timing:
- Elapsed duration = 3.887 s
- Ideal time = 3.223 s
- Compute time = 2.637952 s
- MPI time = 1.24930663 s

Parallel Efficiency:
- Total Efficiency = 0.679
  - Load Balance = 0.950
  - Micro Load Balance = 0.859
  - Transfer = 0.829
  - Bweff = 0.814
  - Leff = 1.000

Load balance:
- Time Load Balance = 0.950
- Instructions Load Balance = 0.990
- Cycles load balance = 0.950
- IPC Load Balance = 0.960

Computational analysis:
- Total useful instructions = 5.399636e+11
- Average useful instructions per process = 8.436931e+09
- Instructions based microload balance/sync = 0.924

Sequential performance:
- Average MIPS = 3199
- Average IPC = 1.050

Basic communication statistics:
  Point to point:
  - Average number of calls = 554.0
  - Balance in number of calls = 1.0
  - Average bytes per process = 728596480.0
  - Balance in bytes = 0.990

Collectives:
  - Num collective calls = 6.0
  - Average bytes per call = 6.670
  - Max bytes per call = 8.000
  - Balance in bytes = 1.000

Inter - Intra node communication statistics:
  Bytes sent (MB):
  - Locally = 22229.811
  - Remotely = 24645.730
  Number of sends:
  - Locally = 3520.000
  - Remotely = 6592.000

64Proc – 8 MPI by node
Performance Analytics
Dominant practice
- We focus a lot on capturing a lot of data
- but we present either everything or first order statistics
- and require new experiments without squeezing the potential information from the previous one

Need for performance analytics
- Leveraging techniques from data analytics, mining, signal processing, life sciences,…
- towards insight
- And models

Some techniques worked on at BSC
- Spectral analysis
- Clustering
- Folding
- Simulation (Dimemas)
Spectral analysis
Applications tend to have Iterative behavior
- Detailed analysis can be applied to a few such iterations

Metrics in Paraver are functions of time
- Natural target for signal processing techniques to automatically detect such iterative structure
- Relevant functions of time at global application level
  - # processes in MPI, outside MPI, …
  - Sum of useful burst duration
    - Semantic: high when many processes are in the middle of very long computation bursts
    - Does capture repetitive structure of application
Techniques
- Mathematical morphology
  - clean up perturbed regions
- Wavelet transform
  - identify coarse regions
- Spectral analysis
  - detailed periodic pattern

Useful
- Identify structure (periodicity)
- Reduce trace sizes
- Increase precision of profiles (report non perturbed stats)
Signal processing applied to performance analysis

Hierarchical structure identification
Scalability: online automatic interval selection

"G. Llort et al, "Scalable tracing with dynamic levels of detail" ICPADS 2011"
Clustering
Clustering: analysis of performance @ serial computation bursts

Identification of computation structure
- CPU burst = region between consecutive runtime (MPI, OpenMP) calls
  - Described with performance hardware counters
  - Associated with call stack data

Scatter plot on some relevant metrics
- Instructions: idea of computational complexity, computational load imbalance,…
- IPC: Idea of absolute performance and performance imbalance
- Automatically Identify clusters
Using Clustering to identify structure

Performance @ serial computation bursts

SPECFEM3D

WRF 128 cores

GROMACS

Asynchronous SPMD

Balanced #instr variability in IPC

SPMD

Repeated substructure

Coupled imbalance

MPMD structure

Different coupled imbalance trends
Example PARSEK (DEEP)

- Duration vs. cluster
- Instruction vs. cluster
How many clusters?
Which is better?

- The two describe interesting structure
- Typically SPMD would be a good first level of description for most apps
Clustering enables focusing the analysis and opens many different uses

- **Analysis**
  - Detection of application structure

- **Precise instantaneous metrics**
  - Correlation of sampled data to generate instantaneous metric evolution

- **Dimemas:**
  - Separate speed factors per cluster on predictive simulations

- **Track the evolution of application behaviour effects**

- ...
Folding
Mixing instrumentation and sampling …

… to get extreme detail with minimal overhead

Different roles
- Instrumentation delimits regions
- Sampling report progress within region

Harald Servat et al. “Detailed performance analysis using coarse grain sampling” PROPER@EUROPAR, 2009

- Instructions evolution for routine copy_faces of NAS MPI BT.B

- Red crosses represent the folded samples and show the completed instructions from the start of the routine.

- Green line is the curve fitting of the folded samples and is used to reintroduce the values into the tracefile.

- Blue line is the derivative of the curve fitting over time (counter rate).
Folding \(\rightarrow\) profiles of rates and ratios

- Call-site sampling information is folded
  - Correlation between hwc and call-sites
  - GVIM/CUBE add-on to show performance within source code
    - Timeless but useful to point performance issues
What is a good performance?

Performance of a sequential region = 2000 MIPS

Is it good enough?

Is it easy to improve?
Instantaneous CPI stack

- Trivial fix (loop interchange)
- Easy to locate?
- Next step?
- Availability of CPI stack models for production processors?
  - Provided by manufacturers?
Fix: Precompute transcendental functions. Use stored value.

Serialization of computation
- vs memory access
  - Interesting tradeoff
Correlating counters

- Between processes
- 3 Algorithmic phases
- Impact of multicore sharing
Correlating counters

- Within a process
- 3 algorithmic phases
- Impact of multicore sharing
Performance analysis tools objective

Help generate hypotheses

Help validate hypotheses

Qualitatively

Quantitatively
Tools: mechanisms and navigation

The tools are instruments to address the questions

Need to know how to use

- First learn to navigate with the tool
  - How to load configurations, zoom, fit coloring scales
  - How to read
  - How to generate timelines form tables

Second

- Develop a basic understanding of the process of generation of the timelines and histograms.

Paraver Tutorial:
Introduction to Analysis with Paraver (MPI)
First steps

Parallel efficiency – percentage of time invested on computation
  – Identify sources for “inefficiency”:
    • load balance
    • Communication /synchronization

Serial efficiency – how far from peak performance?
  – IPC

Scalability – code replication?
  – Total #instructions

Behavioral structure? Variability?

Paraver Tutorial:
Introduction to Paraver and Dimemas methodology
Factors modeling parallel efficiency

- **Load balance (LB)**
- **Communication**
  - Micro load balance ($\mu$LB) or serialization
  - Transfer

Factors describing serial behavior

- **Computational complexity**: $\#\text{instr}$
- **Performance**: IPC

Overall Scaling model

$$\eta = \eta_{LB} \times \muLB \times \text{Transfer}$$

$$\eta_{instr} = \frac{\#\text{instr}_0}{\#\text{instr}_P}$$

$$\eta_{IPC} = \frac{IPC_P}{IPC_0}$$

**CommEff**

Scaling model

\[ \eta = LB \times \text{CommEff} \]

Directly from real execution metrics
Scaling model

Dimemas simulation with ideal target
- Latency = 0; BW = ∞

CommEff = μLB * Transfer

\[ \mu LB = \frac{\text{max}(T_i)}{T_{\text{ideal}}} \]

\[ \text{Transfer} = \frac{T_{\text{ideal}}}{T} \]
Scaling model

Fundamental behavior

Explains bottleneck ...

...how they migrate ...

... and combined effect

\[ \eta \parallel = LB \ast \mu LB \ast \text{Transfer} \]
Modelling efficiency

\[ \eta = LB \cdot \mu LB \cdot \text{Transfer} \]

CG-POP mpi2s1D - 180x120

Good scalability !! Should we be happy?

\[ \eta = \eta \parallel \cdot \eta_{\text{instr}} \cdot \eta_{\text{IPC}} \]
www.bsc.es/paraver

- downloads
  - Sources / Binaries
  - Linux / windows / MAC
- documentation
  - Training guides
  - Tutorial slides

Getting started
- Start wxparaver
- Help → tutorials and follow instructions
- Follow training guides
  - Paraver introduction (MPI): Navigation and basic understanding of Paraver operation
Detailed material
Semantic Module
Basic functions of time

The filter module presents a subset of the trace to the semantic module. Each thread $i$ is described by

- A sequence of events $E_{vi}, i \in N$, states $S_{ti}, i \in N$ and communications $C_{ci}, i \in N$
- For each event let $T(E_{vi})$ be its time and $V(E_{vi})$ its value
- For each state let $T_s(S_{ti})$ be its start time $T_e(S_{ti})$ its stop time and $V(S_{ti})$ its value
- For each Communication let $T_s(C_{ci})$ be its send time, $T_r(C_{ci})$ its receive time, $Sz(C_{ci})$ its size.
- $\text{Partner}(C_{ci})$ and $\text{Dir}(C_{ci}) \in \{\text{send}, \text{recv}\}$ identify the partner process and direction of the transfer

Semantic module builds

$$s(t) = S(i), t \in [t_i, t_{i+1}), i \in N$$

Function of time Series of values
Filter module

Communications that pass through the filter

Events that pass through the filter

Show list of event types
Semantic module: Control
Semantic module

From Events to functions of time

- Last event value
  \[ S(i) = V(E_{v_i}) \]

- Next event value
  \[ S(i) = V(E_{v_{i+1}}) \]

- Average Next Event Value
  \[ S(i) = \frac{V(E_{v_{i+1}})}{T(E_{v_{i+1}}) - T(E_{v_i})} \]

- Interval btw. Events
  \[ S(i) = T(E_{v_{i+1}}) - T(E_{v_i}) \]
Semantic module

From communication records to functions of time

- Send Bytes
  \[ s(t) = \sum_j Sz(C_j), j \mid (T_s(C_j) < t) \land (T_r(C_j) > t) \land (Dir(C_j) = \text{send}) \]

- Send Bandwidth
  \[ s(t) = \sum_j \frac{Sz(C_j)}{T_r(C_j) - T_s(C_j)}, j \mid (T_s(C_j) < t) \land (T_r(C_j) > t) \land (Dir(C_j) = \text{send}) \]

- Msgs in transit
  \[ s(t) = \sum_j sign(j), j \mid (T_s(C_j) < t) \land (T_r(C_j) > t) \land (Dir(C_j) = \text{send}) \]

- Recv. Bandwidth
  \[ s(t) = \sum_j \frac{Sz(C_j)}{T_r(C_j) - T_s(C_j)}, j \mid (T_s(C_j) < t) \land (T_r(C_j) > t) \land (Dir(C_j) = \text{recv}) \]

- Rec. Negative Msgs
  \[ s(t) = \sum_j sign(j), j \mid (T_r(C_j) < t) \land (T_s(C_j) > t) \land (Dir(C_j) = \text{recv}) \]

- Comm. Partner
  \[ s(t) = Partner(C_j), j \mid (T_r(C_j) < t) \land (T_s(C_j) > t) \]

- Bytes btw. Events
  \[ S(i) = \sum_j Sz(C_j), j \mid T_s(C_j) \in [T(Ev_i), T(Ev_{i+1})] \lor T_r(C_j) \in [T(Ev_i), T(Ev_{i+1})] \]
Composition

\[ S'(t) = f(S(t)) \quad \text{and} \quad S' = f \circ S \]

- **Sign**\[ S'(t) = \text{sign}(S(t)) \]
- **1-sign**\[ S'(t) = 1 - \text{sign}(S(t)) \]
- **Select range**\[ S'(t) = S(t) \in [a,b] ? S(t) : 0 \]
- **Sign \° Is equal**\[ S'(t) = \text{sign}(S(t) = a ? S(t) : 0) \]
- **Delta**\[ S'(t) = S_{i+1} - S_i \]
- **Stacked value**
Semantic module

- Derived windows
  - Point wise operation
    - $S = \alpha \cdot S^a <\text{op}> \beta \cdot S^b$
    - $<\text{op}> : +, -, *, /, ...$

L2 Line Loads

L2 miss ratio

Loads
Stores
Mem Ops

x100

L2 miss ratio
Derived windows

- Point wise operation
  - \( S = \alpha * S^a <op> \beta * S^b \)
  - \(<op>: +, -, *, /, \ldots\)

Interval between MPI events

In MPI call

MPI call duration
Semantic module: Examples

Thread function: State as is

- Useful for
  - Global thread activity: computing, idle, fork/join, waiting,…..
Semantic module: Examples

Filter: type == 2
- Thread function: Last event value

- Useful for
  - In parallel region
  - Mutual exclusion
  - Variable values: iteration,....
Semantic module: Examples

Filter: type == 4
- Thread function: Next event value

- Useful for
  - Hwc events (TLB, L1 misses,...) within interval
Semantic module: Examples

Filter: type == 4
  - Thread function: Average next event value

- Useful for
  - Hwc events (TLB, L1 misses,...) per time unit within interval
### Timeline

- **USR_FCT, idA**
- **USR_FCT, idB**
- **USR_FCT, 0**
- **USR_FCT, idB**
- **USR_FCT, idC**
- **USR_FCT, 0**
- **USR_FCT, 0**

### Filter: type == USR_FCT
- Thread function: Last event value
- Compose: Stacked value

### Useful for
- Routine
Semantic module perspective

- **Process model**
  - Thread, task, application, workload

- **Resource model**
  - CPU, node, system

![Process view](image)

![Resource view](image)
**Process model perspective**

- **Semantic value**: $S(t)$

- $S = f_{comp2} \circ f_{comp1} \circ f_{Workload} \circ f_{Application} \circ f_{task} \circ S_{thread}$

- **Semantic functions**
  - $f_{comp2}, f_{comp1}$: sign, mod, div, in range, select range
  - $f_{Application}, f_{Workload}$: add, average, max, select
  - $f_{task}$: add, average, max, select
  - $S_{thread}$: in state, useful, given state,
    - last event value,
    - next event value,
    - average next event value
    - interval between events, …
Resource model perspective

\[ S_{resource} = f_{comp2} \circ f_{comp1} \circ f_{System} \circ f_{Node} \circ f_{CPU} \circ S_{thread} \]

- **Semantic functions**
  - \( f_{System} \): add, average, max, select
  - \( f_{Node} \): add, average, max, select
  - \( f_{CPU} \): active thread, select
  - \( S_{thread} \): in state, useful, given state, next event value, thread_id

![Diagram showing resource model perspective](image-url)
Analysis Module
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 duration, instructions, BW, IPC, ...

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
Single flexible quantitative analysis mechanism

Let
- \(cw_1\) and \(cw_2\) two views we will call control views
- \(dw\) a view we will call data window

For each control window we define a set of bins

\[
bin_{j}^{cw} = \left[ range_{j}^{cw}, range_{j+1}^{cw} \right]
\]

Range

\[
range_{j+1}^{cw} = range_{j}^{cw} + \delta_{cw}
\]

And the discriminator functions

\[
\delta_{j}^{cw}(t) = \left( (S_{cw}(t) \in bin_{j}^{cw}) ? 1 : 0 \right)
\]

Identify regions with \(cw\)'s within the \((j,k)\) bin

\[
\delta_{j,k}(t) = \delta_{j}^{cw}(t) * \delta_{k}^{cw}(t)
\]

The 3D analysis module computes a cube (or plane in the case of 2D) of statistics

\[
M(thread, j, k) = statistic(S_{th}^{dw}(t) * \delta_{th,j,k}(t))
\]

Where the statistic can represent the average value, the number of intervals,....
Distributed Configurations
Distribution of cfg directories

**CFG**

- **General**
  - including basic views (timelines) and analyses (2/3D profiles), including views of the user functions and call-stack

- **Counters_PAPI**
  - Hardware counter derived metrics. Grouped in directories for
    - Program: related to algorithmic/compilation (i.e. instructions, FP ops, …)
    - Architecture: related to execution on specific architectures (i.e. cache misses, …)
    - Performance: metrics reporting rates per time (i.e. MFLops, MIPS, IPC, …)

- **MPI**
  - Grouped in directories displaying views and analysis. Further separated into point to point and collectives.

- **OpenMP**
  - Grouped in directories displaying views and analysis

$\text{PARAVER\_HOME/}\text{cfgs}$
How to …
Main Paraver window

Select to browse in lower panel for traces or cfds

Select to browse characteristics of active view or table

Available views and tables
Active view or table highlighted
Load configuration files

Navigate through directory tree

Select directory

List of directories and configuration files in current directory

Applied to the current tracefile
Shortcuts:

- Drag and move (D&M) – Zoom
- Control D&M – Zoom XY
- Shift D&M – Timing

Hide lower panel (double click)
How to generate table and change statistic

To generate table: click button and select region of the window whose values will determine the columns of the table

Range and bin width (delta) represented by each column. By default is automatically selected, but can be manually changed

Selection of statistic to appear in each cell

Cell coloring gradient control

Window used to compute statistic (only used by some statistics)
3D tables

- One additional dimension
  - One plane per value of a 3D control window
- Useful to categorize histograms
  - i.e. histogram of duration of specific user function

3D control window: determines planes

Actual Plane on display
Table information and control

- Create a new table
- Display whole table / cell text
- Color/not cells
- Transpose
- Hide null columns
- Region analyzed
- Bin definition
- Change Data window
- Activate 3D analysis

- Color encoding
  - max
  - min
**Table information and control**

- **Open Data window**
- **Open Control window**
- **Open 3D window**
- **Generate a timeline, derived from the control window with the range of values selected clicking in the table (zoom mode only)**
- **Right click**
- **Generate ASCII file with table data**
- **Shortcuts (zoom mode only):**
  - Drag and move (D&M) – Zoom
  - Control D&M – Zoom XY

---

**Selected plane**

---

**Copy**

- Paste
- Paste Special...

**Paste Special**

- Time
- Objects
- Size
- Semantic Scale
- Control scale
- 3D scale

---

**Linear**

- Steps
- Logarithmic
- Exponential
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?

Click button and select column(s)

Will automatically generate derived views from the global view

Only showing when is routine white executing

Only showing when is routine pink executing
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?

Click button and select column(s)

Will automatically generate

3D histogram of duration of routine foo

Only showing duration of routine foo
Trace manipulation
Handling very large traces

Paraver data handling utilities

- If trying to load a very large trace, Paraver will ask if you want to filter it

Three steps:

- Filter original trace discarding most of the records only keeping most relevant information (typically computation bursts longer than a given lower bound)
- Analyze coarse grain structure of trace. Typically `useful_duration.cfg`
- Cut original trace to obtain a fully detailed trace for the time interval considered representative or of interest

Guided hands-on available in

http://www.bsc.es/computer-sciences/performance-tools/documentation ➔ Trace Preparation
Filtering very large traces

Trace to which it will be applied

A trace with basename.filter1.prv will be generated

Discard events and communications

Keep only Running bursts

--- longer than 3000 ns
Cutting very large traces

Load a filtered trace and use the scissors tool

Browse to select file from which the cut will be obtained

Select cutter

Click to select region

Select time interval by clicking left and right limits in a window of the filtered trace previously loaded

Recommended cuts within long computation bursts

Default setups
#!/bin/bash

export NP=8
export INPUT=$1

cleo-submit -np $NP ./HydroC -i $INPUT
Adapt job submission script

```
#!/bin/bash

export NP=8
export INPUT=$1

cleo-submit -np $NP ./trace.sh ./HydroC -i $INPUT
```

```
#!/bin/bash

export EXTRA_E_HOME=/export/hopsa/BSCTools/tools/extrae-2.3
export EXTRA_CONFIG_FILE=extrae/extrae.xml

export LD_PRELOAD=$EXTRA_E_HOME/lib/libmpitrace.so

export EXE=$1
export TRACENAME=${EXE}_${3}.prv

$@
```

**appl.job**

**trace.sh**
Trace control .xml

```
<?xml version='1.0'?><trace enabled="yes"
    home="/home/judit/tools/extrae-2.3"
    initial-mode="detail"
    type="paraver"
    xml-parser-id="Id: xml-parse.c 799 2011-10-20 16:02:03Z harald $"
>
<mpi enabled="yes">
    <counters enabled="yes" />
</mpi>

<openmp enabled="no">
    <locks enabled="no" />
    <counters enabled="yes" />
</openmp>

<callers enabled="yes">
    <mpi enabled="yes">1-3</mpi>
    <sampling enabled="no">1-5</sampling>
</callers>
```

- Activate MPI tracing and emit hardware counters at MPI calls
- Do not activate OpenMP tracing
- Emit call stack information (number of levels) at acquisition points

Details in $EXTRAE_HOME/share/example/MPI/extrae_explained.xml
Trace control .xml (cont)

extrae.xml (cont)

```xml
<user-functions enabled="no" list="/home/bsc41/bsc41273/user-functions.dat">
  <max-depth enabled="no">3</max-depth>
  <counters enabled="yes" />
</user-functions>
```

Add instrumentation at specified user functions
Requires Dyninst based mpitrace
Trace control .xml (cont)

...
<storage enabled="no">
  <trace-prefix enabled="yes">TRACE</trace-prefix>
  <size enabled="no">5</size>
  <temporal-directory enabled="yes" make-dir="no">/scratch</temporal-directory>
  <final-directory enabled="yes" make-dir="no">/gpfs/scratch/</final-directory>
  <gather-mpits enabled="no" />
</storage>

Size of in core buffer (#events)

... name, tmp and final dir ...

... max (MB) per process size (stop tracing when reached)

mpitrace.xml (cont)
Trace control .xml (cont)

...<trace-control enabled="yes">
  <file enabled="no" frequency="5m">/gpfs/scratch/bsc41/bsc41273/control</file>
  <global-ops enabled="no"></global-ops>
  <remote-control enabled="no">
    <signal enabled="no" which="USR1"/>
  </remote-control>
</trace-control>

...<others enabled="no">
  <minimum-time enabled="no">10M</minimum-time>
  <terminate-on-signal enabled="no">USR2</terminate-on-signal>
</others>

...
Trace control .xml (cont)

```xml
<bursts enabled="no">
  <threshold enabled="yes">500u</threshold>
  <counters enabled="yes" />
  <mpi-statistics enabled="yes" />
</bursts>

<sampling enabled="no" type="default" period="5m" />

... emit only computation bursts of a minimal duration ...
... plus summarized MPI events

Activate/not time based sampling and how often
```
... 

<merge enabled="yes" synchronization="default" binary="$EXE$" tree-fan-out="16" max-memory="512" joint-states="yes" keep-mpits="yes" sort-addresses="yes" >

$TRACENAME$

</merge>

</trace>

**mpitrace.xml (cont)**

**Merge individual traces into global application trace at end of run ...**

**... into this trace name**
Library depends on programming model

<table>
<thead>
<tr>
<th>Programming model</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>libseqtrace</td>
</tr>
<tr>
<td>Pure MPI</td>
<td>libmpitrace[f]¹</td>
</tr>
<tr>
<td>Pure OpenMP</td>
<td>libomptrace</td>
</tr>
<tr>
<td>Pure Pthreads</td>
<td>libpttrace</td>
</tr>
<tr>
<td>CUDA</td>
<td>libcudatrace</td>
</tr>
<tr>
<td>MPI + OpenMP</td>
<td>libompitrace[f]¹</td>
</tr>
<tr>
<td>MPI + Pthreads</td>
<td>libptmpitrace[f]¹</td>
</tr>
<tr>
<td>Mpi + CUDA</td>
<td>libcudampitrace[f]¹</td>
</tr>
</tbody>
</table>

¹ for Fortran codes
Scalability
Linpack @ Marenostrum: 10k cores x 1700 s

- Dgemm duration: 11.8 s
- Dgemm duration: 10 s
- Dgemm IPC: 2.95
- Dgemm IPC: 2.85
- Dgemm L1 miss ratio: 0.8
- Dgemm L1 miss ratio: 0.7
Scalability of analysis

Jugene

~ 105 seconds

8K cores

12K cores

16K cores

Jaguar

~ 47 seconds

Flow Tran

Tran Flow

PFLOTRAN
Data reduction techniques

Software counters
- Summarize information of some event types (i.e. MPI calls) by emitting aggregate counts
- Emit counts at structurally relevant points (i.e. begin and end of long computation phases)

Representative cuts
- Emit full detail only on selected intervals, representative of full program execution

On and off line combinations
- By instrumentation
- By paraver filtering

Software counters

- **Useful duration**
- **% MPI time**
- **# collectives**
- **Collective bytes**
- **# p2p**
- **p2p bytes**
- **p2p BW**

**GADGET, PRACE Case A, 1024 procs**
Software counters

GADGET, PRACE Case A, 2048 procs
Software counters

![Graph showing Speedup vs. processors]

**Useful duration**

**% MPI time**

**# collectives**

**Collective bytes**

**# p2p**

**p2p bytes**

**p2p BW**

GADGET, PRACE Case A, 4096 procs