Challenges on the way to Exascale

- Efficiency (..., power, ...)  
- Variability  
- Memory  
- Faults  
- Scale (..., concurrency, strong scaling,...)  
- Complexity (..., Hierarchy/Heterogeneity,...)
Key issues

- Programming model and run time
  - Language to express ideas
  - Runtime responsible for mapping them to resources
  - Smooth transition path, long lived efforts

- Performance analysis tools
  - Avoid flying blind
  - The importance of details
  - How can we improve?
  - Need much more "Performance analytics"

- Algorithms
  - Final target
  - A say in the solution

Outline

- Introduction
- Paraver
  - General description
  - Finding needles in haystacks
  - Scalability
  - Heterogeneous support
- Dimemas
- Scaling model
- Structure detection
- HWC analyses
  - Projection and CPI Stack models
  - Folding: Instrumentation + sampling
- Tareador
- Conclusions
Performance Tools

Why Tools?

- Measurement techniques as enablers of science

- Are becoming vital for program development at exascale
  - Fly with instruments: survive the unexpected, ...  
  - Not to shoot blind: Work in the right direction

- Are important for "Lawyers"
  - Know who to blame

- Are vital for system architects
  - Understand our systems, productivity

- Performance analyst:
  - A specialist understanding displays
Science

- printf()
- timers

Advances linked to capability of observation and measurement

Computer Science

Performance analysis:
- A lot of speculation
  - We see $\int_a^b f(t) dt$
  - We talk about $f(t)$

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
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 framework

Trace handling & display
Simulators
Analytics

Open Source
(Linux and windows)
http://www.bsc.es/paraver

The importance of detail and intelligence
Paraver

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
What is Paraver

- A browser …
- … for sequences of timestamped events …
- … with a multispectral philosophy …
- … and a mathematical foundation …
- … that happens to be mainly used for performance analysis

Paraver mathematical foundation

- Every behavioral aspect/metric of a thread can be described as a function of time
  - Need a language to describe how to compute such functions of time.
    - Basic operators (from) trace records
    - Ways of combining them
  - Aggregation along the process model (thread, process, application, workload) or resource model (core, node, system) can be done to obtain a function of time at the corresponding level

- 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
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
  - State, user function, outlined routine
    \[ S_i \in [0, n] \subseteq N, \quad n < \]
- Logical
  - In specific user function, In MPI call, In long MPI call
    \[ S_i \in \{ 0, 1 \} \]
- Numerical
  - IPC, L2 miss ratio, Duration of MPI call, duration of computation burst
    \[ S_i \in R \]

**Timelines**

- **Representation**
  - Function of time
  - Color encoding
  - Gradient color
    - Light green → Dark blue
  - Not null gradient
    - Black for zero value
    - Light green → Dark blue

- **Non linear rendering to address scalability**
Basic functions of time

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

- A sequence of events $Ev_i, i \in N$, states $St_i, i \in N$ and communications $C_i, i \in N$.
- For each event let $T(Ev_i)$ be its time and $V(Ev_i)$ its value.
- For each state let $T_i(St_i)$ be its start time, $T_i(St_i)$ its stop time and $V(St_i)$ its value.
- For each Communication let $T_i(C_i)$ be its send time, $T_i(C_i)$ its receive time, $St(C_i)$ its size.
- Partner($C_i$) and Dir($C_i$) 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
Tables

Huge number of statistics computed from timelines

- Useful Duration
- IPC
- Instructions
- L2 miss ratio
Tables: Profiles, histograms, correlations

By the way: six months later ….

Useful Duration
IPC
Instructions
L2 miss ratio

Paraver – Performance data browser

Raw data
Timelines
2/3D tables (Statistics)

Goal = Flexibility
No semantics
Programmable

Configuration files
Distribution
Your own

Comparative analyses
Multiple traces
Synchronize scales
Paraver: finding needles in haystacks

From tables to timelines

Where in the timeline do the values in certain table columns appear?
- i.e. 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

Scalability
Scalability of Presentation

Linpack @ Marenostrum: 10k cores x 1700 s

<table>
<thead>
<tr>
<th>Dgemm duration</th>
<th>Dgemm IPC</th>
<th>Dgemm L1 miss ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.8 s</td>
<td>2.95</td>
<td>0.8</td>
</tr>
<tr>
<td>10 s</td>
<td>2.85</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Scalability of analysis

Jugene

- ~ 105 seconds
- 8K cores
- 12K cores
- 16K cores

Jaguar

- ~ 47 seconds
- 8K cores
- 12K cores
- 16K cores

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
Software counters

GADGET, PRACE Case A, 2048 procs

GADGET, PRACE Case A, 4096 procs
Scalability: Data reduction

- Data handling/summarization capability
  - Software counters, filtering and cutting
  - Supported by GUI.

- Automatizable through signal processing techniques:
  - Mathematical morphology to clean up perturbed regions
  - Wavelet transform to identify coarse regions
  - Spectral analysis for detailed periodic pattern

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
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

Analyze coarse grain structure

- Filtered trace is a Paraver trace
- Can be analyzed with standard cfgs as long as the information they require is still in the trace
  - A typical view that shows a lot of the structure of a trace is useful_duration.cfg
  - Repetitive structure is often apparent
  - Perturbations can also be typically identified
  - A clean/representative interval can be identified
**Cutting very large traces**

Load a filtered trace and use the scissors tool

- **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

**Heterogeneous systems**

- **GPUs**
- **KNC:**
  - OmpSs @ KNC ok.
  - Still to be ported to intel accelerator model

- **OmpSs**
  - Cholesky n = 8192; bs = 1024

- **HMPP**
Dimemas

BSC – tools framework

Predictions/expectations

Extrae
- MRNET
- Valgrind
- Dyninst
- PAPI

Paraver

Paramedir

DIMEMAS
- VENUS (IBM-ZRL)

Performance analytics

CUBE, grpupil, vi...

Machine description

Instruction level simulators

Time Analysis, filters

.prv

.par

.prv2trf

.DIMEMAS

.XML

.control

.XML

Control

Predictions/expectations

사를.

.BSC – tools framework

Dimension

Machine description

Performance analytics

Prediction/expectations

_machine

XML

.Control

Predictions/expectations

.machines

.XML
Dimemas: Coarse grain, Trace driven simulation

Simulation: Highly non linear model

- Linear components
  - Point to point communication
  - Sequential processor performance
    - Global CPU speed
    - Per block/subroutine

- Non linear components
  - Synchronization semantics
    - Blocking receives
    - Rendezvous
  - Resource contention
    - CPU
    - Communication subsystem
      » links (half/full duplex), busses

P2P communication model

Early receiver

MPI_send

Logical Transfer

MPI_recv

Physical Transfer

Size

BW

Process Blocked

Machine Latency

Uses CPU

Independent of size

Computation proceeds

Simulated contention for machine resources

(links & busses)
Intrinsic application behavior

Load balanced and dependence problems?
- \( BW = \infty, \quad L = 0 \)

- \textbf{GADGET @ Nehalem cluster}
  - 256 processes

- Real run
- Ideal network

End point contention
- Simulation with Dimemas
  - Very low BW
  - 1 output link, \( \infty \) input links

Recommendation:
- Important to schedule communications.

Everybody sending by destination rank order
Endpoint contention at low ranked processes
Impact of architectural parameters

- Ideal speeding up ALL the computation bursts by the CPU ratio factor
- The more processes the less speedup (higher impact of bandwidth limitations) !!!!
Scaling model

Presenting application performance

Factors modeling parallel efficiency
- Load balance (LB)
- Micro load balance (μLB) or serialization
- Transfer

\[ \eta = LB \times \muLB \times \text{Transfer} \]

Factors describing serial behavior
- Performance: IPC

Scaling model

\[ \text{Sup} = \frac{P}{P_0} \times \frac{\eta}{\eta_0} \times \frac{\text{IPC}}{\text{IPC}_0} \times \frac{\#\text{instr}_0}{\#\text{instr}} \]
Parallel Performance Models

\[ \eta = LB \times \muLB \times \text{Transfer} \]

\[ T_{\text{Comp}} = \frac{\#\text{instr}}{IPC} \]

*Old v new version: -43% instr, -52% time*

Scaling model

Sup = \frac{P \times LB \times \text{CommEff}}{P_0 \times LB_0 \times \text{CommEff}_0} \times \frac{\text{IPC} \times \#\text{instr}}{\text{IPC}_0 \times \#\text{instr}_0}

Directly from real execution metrics

\[ \text{Side effect} = \frac{\text{Efficiency}}{\text{IPC} \times \text{Instr}} \]

\[ \text{LB} = \frac{\sum\text{Eff}}{P \times \text{max} \text{Eff}} \]

\[ \text{microLB} = \max \left( \frac{T_{\text{Ideal}}}{T_{\text{Inst}}} \right) \]

Requires Dimemas simulation

Migrating/local load imbalance
Serialization

\[ T_{\text{Comp}} = \frac{\text{max}(T_i)}{T_{\text{Inst}}} \]
Performance Analytics

Automatic analysis: Timeline Structure

- Automatic Performance Analysis:
  - Focus analyst's work to relevant regions
  - Report non perturbed stats
  - Suggest fixes

- Timeline analysis
  - Hierarchical structure identification
  - Spectral analysis, mathematical morphology,…
Signal processing applied to performance analysis

- Identify structure
- Reduce trace sizes
- Increase precision of profiles

Flushing
Flushing filtered
Wavelet
High frequency

∑ Useful Duration
Spectral density
Autocorrelation

Scalability: online automatic interval selection

Back-end threads
Front-end threads
Back-end threads

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

- Burst = continuous computation region
  - between exit of an MPI call and entry to the next
  - Instrumented routine

- 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

HWC projection

- Complete hardware counter characterization
  - Precise Statistics (>> multiplexing)
  - CPI stack model

- From a single run

<table>
<thead>
<tr>
<th>Cluster</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Time</td>
<td>51.88</td>
<td>17.95</td>
<td>16.80</td>
<td>6.44</td>
<td>1.42</td>
</tr>
<tr>
<td>Avg. Burst Dur. (ns)</td>
<td>1.02</td>
<td>0.78</td>
<td>13.14</td>
<td>2.50</td>
<td>1.11</td>
</tr>
<tr>
<td>IPC</td>
<td>1.02</td>
<td>0.65</td>
<td>0.89</td>
<td>0.91</td>
<td>0.53</td>
</tr>
<tr>
<td>MIPS</td>
<td>2231.4</td>
<td>1421.5</td>
<td>1006.5</td>
<td>201.5</td>
<td>1165.0</td>
</tr>
<tr>
<td>MFLOPS</td>
<td>819.2</td>
<td>66.3</td>
<td>140.2</td>
<td>260.2</td>
<td>23.6</td>
</tr>
<tr>
<td>L1M/Kbyte</td>
<td>0.02</td>
<td>1.53</td>
<td>1.19</td>
<td>1.17</td>
<td>2.68</td>
</tr>
<tr>
<td>L2M/Kbyte</td>
<td>0.06</td>
<td>1.26</td>
<td>0.99</td>
<td>0.35</td>
<td>0.21</td>
</tr>
<tr>
<td>Mem.BW (MB/s)</td>
<td>16.79</td>
<td>218.17</td>
<td>13.87</td>
<td>85.77</td>
<td>29.76</td>
</tr>
</tbody>
</table>

J. Gonzalez et al., “Performance Data Extrapolation in Parallel Codes”, ICPPDS 2010
Automatic clustering quality assessment

- Leverage Multiple Sequence alignment tools from Life Sciences
- Process == Sequence of clusters  <-> sequence of amino acids == DNA
- CLUSTAL W, T-Coffee, Kalign2
- Cluster Sequence Score (0..1)
- Per cluster / Global
  - Weighted average

Folding: Instrumentation + sampling

- Extremely detailed time evolution of hardware counts, rates and callstack
- Minimal overhead
- Based on
  - Instrumentation events (iteration, MPI, ...) and periodic samples.
  - Application structure: manual iteration instrumentation, routines, clusters
- Folding
  - Post processing to project all samples into one instance

Harald Servat et al. “Detailed performance analysis using coarse grain sampling” PROPER@EUROPAR, 2009
Folding → profiles of rates


Clustering + Folding + CPI Stack

- Unmodified production binary
  - Instrumentation of MPI calls plus periodic samples.
  - Rotating hardware counters.
  - Clustering detects structure to enable folding
  - Folding of multiple hardware counters to compute CPI stack model

- Instantaneous MIPS and CPI Stack model
  - Clean abstract identification of performance problem
  - Good identification of phases

CGPOP
Interchanging loops (MR. GENESIS)

Pre-computing some data – loop split (PMEMD)
Tareador

Predicting performance

- Performance prediction
  - Predict MPI/StarSs multithreaded from pure MPI
  - Leveraging other tools in environment

Diagram:
- Input code
  - Code translation
  - mpicc
  - MPI process
  - Valgrind tracer
  - Potential (MPI+SMPS)

Incomplete/suggested taskification
- pragmas
- runtime calls (even on not well structured code)
Predicting performance

- Potential concurrency between task (for a suggested taskification)...

- ... as number of cores increases.
Conclusion

- Extreme flexibility:
  - Maximize iterations of the hypothesis – validation loop
  - Learning curve
  - “Don’t ask whether something can be done, ask how can it be done”

- Detailed and precise analysis
  - Squeeze the information obtainable form a single run
  - Insight and correct advise with estimates of potential gain

- Data analysis techniques applied to performance data
Tools web site

- www.bsc.es/paraver
  - downloads
    - Sources
    - Binaries
  - documentation
    - Training guides (Documentation → Paraver Introduction (MPI))
    - Tutorial slides

An analogy

Use of traces
Huge probe effect
Team work

Multidisciplinary
Correlate different sources
Speculate till arriving to consistent theory