Understanding applications with Paraver
tools@bsc.es
Our Tools

- Since 1991
- Based on traces
- Open Source
- [http://tools.bsc.es](http://tools.bsc.es)

Core tools:

- Paraver (paramedir) – offline trace analysis
- Dimemas – message passing simulator
- Extrae – instrumentation

Focus

- Detail, variability, flexibility
- Behavioral structure vs. syntactic structure
- Intelligence: Performance Analytics
Paraver
Paraver – Performance data browser

- Raw data
- Trace visualization/analysis + trace manipulation
- Timelines
- 2/3D tables (Statistics)
- Goal = Flexibility
  - No semantics
  - Programmable
- Comparative analyses
  - Multiple traces
  - Synchronize scales
From timelines to tables

MPI calls profile

Histogram Useful Duration
Analyzing variability

Useful Duration

IPC

Instructions

L2 miss ratio
Analyzing variability

- By the way: six months later ....
From tables to timelines

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?

[Image of histograms and timelines]
Trace manipulation

- **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, ...

Diagram:
- Initial trace: 570 s, 2.2 GB, MPI, HWC
- Filtered trace: 570 s, 5 MB
- Final trace: 4.6 s, 36.5 MB
- WRF-NMM
  - Peninsula 4km
  - 128 procs

---

Barcelona Supercomputing Center
Centro Nacional de Supercomputación
Extrae
Extrae features

• Platforms
  • Intel, Cray, BlueGene, MIC, ARM, Android, Fujitsu Sparc...

• Parallel programming models
  • MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, Java, Python...

• Performance Counters
  • Using PAPI interface

• Link to source code
  • Callstack at MPI routines
  • OpenMP outlined routines
  • Selected user functions (Dyninst)

• Periodic sampling

• User events (Extrae API)
# Extrae overheads

<table>
<thead>
<tr>
<th></th>
<th>Average values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>150 – 200 ns</td>
</tr>
<tr>
<td>Event + PAPI</td>
<td>750 ns – 1.5 us</td>
</tr>
<tr>
<td>Event + callstack (1 level)</td>
<td>1 us</td>
</tr>
<tr>
<td>Event + callstack (6 levels)</td>
<td>2 us</td>
</tr>
</tbody>
</table>
How does Extrae work?

• Symbol substitution through LD_PRELOAD
  • Specific libraries for each combination of runtimes
    • MPI
    • OpenMP
    • OpenMP+MPI
    • ...

• Dynamic instrumentation
  • Based on Dyninst (developed by U.Wisconsin / U.Maryland)
    • Instrumentation in memory
    • Binary rewriting

• Alternatives
  • Static link (i.e., PMPI, Extrae API)
Extrae XML configuration

```xml
<mpi enabled="yes">
    <counters enabled="yes" />
</mpi>

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

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

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

Trace the MPI calls
(What’s the program doing?)

Trace the call-stack
(Where in my code?)
Extrae XML configuration (II)

```xml
<counters enabled="yes">
  <cpu enabled="yes" starting-set-distribution="1">
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_L1_DCM, PAPI_L2_DCM, PAPI_L3_TCM
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_BR_MSP, PAPI_BR_UCN, PAPI_BR_CN, RESOURCE_STALLS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_VEC_DP, PAPI_VEC_SP, PAPI_FP_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_LD_INS, PAPI_SR_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, RESOURCE_STALLS:LOAD, RESOURCE_STALLS:STORE, RESOURCE_STALLS:ROB_FULL, RESOURCE_STALLS:RS_FULL
    </set>
  </cpu>
  <network enabled="no" />
  <resource-usage enabled="no" />
  <memory-usage enabled="no" />
</counters>
```

Select which HW counters are measured (How’s the machine doing?)
Extrae XML configuration (III)

```xml
<buffer enabled="yes">
    <size enabled="yes">500000</size>
    <circular enabled="no" />
</buffer>

<sampling enabled="no" type="default" period="50m" variability="10m" />

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

- **Trace buffer size** (Flush/memory trade-off)
- **Enable sampling** (Want more details?)
- **Automatic post-processing to generate the Paraver trace**
Dimemas
Dimemas – Coarse grain, Trace driven simulation

- Simulation: Highly non linear model
  - MPI protocols, resource contention...

- Parametric sweeps
  - On abstract architectures
  - On application computational regions

- What if analysis
  - Ideal machine (instantaneous network)
  - Estimating impact of ports to MPI+OpenMP/CUDA/...
  - Should I use asynchronous communications?
  - Are all parts equally sensitive to network?

- MPI sanity check
  - Modeling nominal

- Paraver – Dimemas tandem
  - Analysis and prediction
  - What-if from selected time window

Detailed feedback on simulation (trace)
Network sensitivity

- MPIRE 32 tasks, no network contention

- $L = 5\mu s$ – $BW = 1\text{ GB/s}$

- $L = 1000\mu s$ – $BW = 1\text{ GB/s}$

- $L = 5\mu s$ – $BW = 100\text{ MB/s}$

All windows same scale
Network sensitivity

- WRF, Iberia 4Km, 4 procs/node
  - Not sensitive to latency
- NMM
  - BW – 256MB/s
  - 512 – sensitive to contention
- ARW
  - BW - 1GB/s
  - Sensitive to contention
Would I benefit from asynchronous communications?

**SPECFEM3D**

Real

Ideal

Prediction

MN

Prediction 100MB/s

Prediction 10MB/s

Prediction 5MB/s

Prediction 1MB/s

Courtesy Dimitri Komatitsch
Ideal machine

The impossible machine: \( BW = \infty, \quad L = 0 \)

- Actually describes/characterizes Intrinsic application behavior
  - Load balance problems?
  - Dependence problems?

GADGET @ Nehalem cluster
256 processes

Real run

Ideal network

Impact on practical machines?
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)!!
Hybrid parallelization

- Hybrid/accelerator parallelization
  - Speed-up SELECTED regions by the CPU ratio factor

(Previous slide: speedups up to 100x)
Efficiency Models
Parallel efficiency model

- Parallel efficiency = LB eff * Comm eff
Parallel efficiency refinement:
$LB \times \muLB \times Tr$

- Serializations / dependences ($\muLB$)
- Dimemas ideal network $\rightarrow$ Transfer (efficiency) = 1
Why scaling?

\[ \eta = LB \times Ser \times Trf \]

Good scalability!!
Should we be happy?

CG-POP mpi2s1D - 180x120

\[ \text{speed up} \]

\[ \text{Efficiency} \]

Parallel eff
LB
uLB
transfer

Parallel eff
Instr. eff
IPC eff
### Some examples of efficiencies

<table>
<thead>
<tr>
<th>Code</th>
<th>Parallel efficiency</th>
<th>Communication efficiency</th>
<th>Load Balance efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gromacs@mt</td>
<td>66.77</td>
<td>75.68</td>
<td>88.22</td>
</tr>
<tr>
<td>BigDFT@altamira</td>
<td>59.64</td>
<td>78.97</td>
<td>75.52</td>
</tr>
<tr>
<td>CG-POP@mt</td>
<td>80.98</td>
<td>98.92</td>
<td>81.86</td>
</tr>
<tr>
<td>ntchem_mini@pi</td>
<td>92.56</td>
<td>94.94</td>
<td>97.49</td>
</tr>
<tr>
<td>nicam@pi</td>
<td>87.10</td>
<td>75.97</td>
<td>89.22</td>
</tr>
<tr>
<td>cp2k@jureca</td>
<td>75.34</td>
<td>81.07</td>
<td>92.93</td>
</tr>
<tr>
<td>icon@mistral</td>
<td>79.86</td>
<td>84.02</td>
<td>95.05</td>
</tr>
<tr>
<td>k-Wave@salomon</td>
<td>89.08</td>
<td>92.84</td>
<td>95.96</td>
</tr>
<tr>
<td>fleur@claix</td>
<td>76.22</td>
<td>90.66</td>
<td>84.07</td>
</tr>
</tbody>
</table>
## Same code, different behaviour

<table>
<thead>
<tr>
<th>Code</th>
<th>Parallel efficiency</th>
<th>Communication efficiency</th>
<th>Load Balance efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>lulesh@mn3</td>
<td>90.55</td>
<td>99.22</td>
<td>91.26</td>
</tr>
<tr>
<td>lulesh@lefraru</td>
<td>69.15</td>
<td>99.12</td>
<td>69.76</td>
</tr>
<tr>
<td>lulesh@uv2 (mpt)</td>
<td>70.55</td>
<td>96.56</td>
<td>73.06</td>
</tr>
<tr>
<td>lulesh@uv2 (impi)</td>
<td>85.65</td>
<td>95.09</td>
<td>90.07</td>
</tr>
<tr>
<td>lulesh@mt</td>
<td>83.68</td>
<td>95.48</td>
<td>87.64</td>
</tr>
<tr>
<td>lulesh@cori</td>
<td>90.92</td>
<td>98.59</td>
<td>92.20</td>
</tr>
<tr>
<td>lulesh@thunderX</td>
<td>73.96</td>
<td>97.56</td>
<td>75.81</td>
</tr>
<tr>
<td>lulesh@jetson</td>
<td>75.48</td>
<td>88.84</td>
<td>84.06</td>
</tr>
<tr>
<td>lulesh@claix</td>
<td>77.28</td>
<td>92.33</td>
<td>83.70</td>
</tr>
<tr>
<td>lulesh@jureca</td>
<td>88.20</td>
<td>98.45</td>
<td>89.57</td>
</tr>
<tr>
<td>lulesh@mn4</td>
<td>86.59</td>
<td>98.77</td>
<td>87.67</td>
</tr>
<tr>
<td>lulesh@inti</td>
<td>88.16</td>
<td>98.65</td>
<td>89.36</td>
</tr>
</tbody>
</table>

**Warning::: Higher parallel efficiency does not mean faster!**
Analytics
Using Clustering to identify structure

Automatic Detection of Parallel Applications Computation Phases (IPDPS 2009)
What should I improve?

What if ....

... we increase the IPC of Cluster1?

... we balance Clusters 1 & 2?

PEPC

13% gain

19% gain
Tracking scalability through clustering

- OpenMX (strong scale from 64 to 512 tasks)
Tracking scalability through clustering

- OpenMX (strong scale from 64 to 512 tasks)
Folding

- Instantaneous metrics with minimum overhead
  - Combine instrumentation and sampling
    - Instrumentation delimits regions (routines, loops, ...)
    - Sampling exposes progression within a region
  - Captures performance counters and call-stack references
“Blind” optimization

- From folded samples of a few levels to timeline structure of “relevant” routines
CG-POP multicore MN3 study

- Unbalanced MPI application
- Same code
- Different duration
- Different performance
Methodology
Performance analysis tools objective

Help generate hypotheses

Help validate hypotheses

Qualitatively

Quantitatively
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, correlate with other counters

• Scalability – code replication?
  • Total #instructions

• Behavioral structure? Variability?

Paraver Tutorial:
Introduction to Paraver and Dimemas methodology
BSC Tools web site

• tools.bsc.es
  • 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