Objective:
- Identify performance problems and help optimize application

Phases
- Data acquisition
- Processing
  - Compaction
  - Summarization: Statistics
- Presentation
  - Textual
  - Graphical
  - Esoteric …
Performance tools

Data acquisition

- Insert probes into the running program
- Issues
  - Control flow: when probe is called
  - Information available to the probe
Data acquisition

**Control flow**
- **Sampling**
  - Punctual samples of program activity.
  - Interrupt driven: correlated or not with program activity.
  - Need to project samples to total program behavior.
- **Instrumentation**
  - Instrument every occurrence of relevant events.
  - How:
    - Static: link with special instrumentation library
      » PMPI
    - Dynamic: binary modification at run time
      » Dyninst, DPCL

**Information available**
- On control flow
  - PC, call stack.
- Arguments to control flow point
  - Example: access to the MPI call arguments in PMPI.
- Additional interfaces
  - Information on internal events of much finer granularity. Accumulated by an external monitoring mechanism.
  - Example:
    - Timing
    - Hardware counters (PAPI, PMAPI,...)
    - OS (rusage)
    - PERSUSE
Data acquisition

- Perturbation:
  - Probe effect
  - f (granularity, overhead)
    - Granularity
      - Program
      - What to instrument
    - Overhead
      - Control flow
      - Time measurement
      - Inline processing
      - Storage to buffer
        » Written to disk when full

Presentation

- How
  - Textual
  - Graphical

- Type
  - Profile
    - Accumulated statistics
      - time, event counts, hardware counter metrics,…
    - Per program component
      - line, function inclusive, function exclusive, process,…
  - Timeline:
    - Instantaneous value of metric vs. time
    - Per process
Keep in mind objective

- Maximize flow of information to user
  - Qualitative
    - Colors, shapes, ...
  - Quantitative
    - Numbers
- by a proper balance of approaches

Philosophies: where processing goes

Typical approaches

- Compute statistics in line to generate a small dump at the end of the run. Minimal computation in display phase
  - Examples: typical profilers
- Dump information to a trace. Postpone most of the processing to the display phase.
  - Example: trace visualization tools

Other approaches

- Dump information to a trace. Perform an elaborated processing to extract information from the trace. Minimal computation in display phase
  - Example: EXPERT
Philosophies: where processing goes

Issues for the user

- Do I need to build special binaries?
  - Modify the source code? How much?
  - Special compilation flags?
  - Link with special libraries?

  ... the less the better

- Do I need to handle a lot of intermediate data?
  - Trace files,…

  ... the less the better
**Issues for the user**

- How fast do I get useful information
  - Relation to source code
    - ... the faster the better

- How detailed/deep is the information I get

---

**Methodology**

- Easier / faster first ...
  - Profile
  - If behavior not easily understandable

- ...then go to details ...
  - Trace visualization and analysis will support
    - Observe time dependence of behavior
    - Look at a richer set of statistics
    - And further
      - Observe variance (time and space)
      - Obtain new performance indices and statistics
    - ... because insight often grows in the details
Profiling @ SGI

- **Instrumentation program**
  - Ssrun: sampling every
    - Application time (user+system)
    - User time
    - Number of instructions, cache misses,...

- **Post processing / presentation**
  - Cvperf: graphical
  - Prof: textual

```bash
karnak 93% ssrun -usertime ./LUBb
karnak 94% prof LUBb.usertime.m6877966 > seq.usertime.txt
karnak 95% prof -b LUBb.usertime.m6877966 > seq.usertime.b.txt

karnak 98% ssrun -pcsamp ./LUBb
karnak 103% prof -l LUBb.pcsamp.m6437465 > seq.pcsamp.h.txt

karnak 113% ssrun -exp dc_hwc ./LUBb
```
SpeedShop profile listing generated Mon May 27 12:41:38 2002
prof LUBb.usertime.m6877966
LUBb (n64): Target program
usertime: Experiment name
ut:cu: Marching orders
R10000 / R10010: CPU / FPU
64: Number of CPUs
250: Clock frequency (MHz.)

Experiment notes--
From file LUBb.usertime.m6877966:
Caliper point 0 at target begin, PID 6877966
/LUBb
Caliper point 1 at exit(0)

Summary of statistical callstack sampling data (usertime)---
154: Total Samples
0: Samples with incomplete traceback
4.620: Accumulated Time (secs.)
30.0: Sample interval (msecs.)

Statistical significance?

Function list, in descending order by exclusive time

<table>
<thead>
<tr>
<th>Index</th>
<th>excl.secs</th>
<th>excl.%</th>
<th>cum.%</th>
<th>incl.secs</th>
<th>incl.%</th>
<th>samples</th>
<th>procedure (dso: file, line)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4]</td>
<td>4.300</td>
<td>97.4%</td>
<td>97.4%</td>
<td>4.300</td>
<td>97.4%</td>
<td>150</td>
<td>bmod (LUBb: LUBb.f, 122)</td>
</tr>
<tr>
<td>[5]</td>
<td>0.060</td>
<td>1.3%</td>
<td>98.7%</td>
<td>0.060</td>
<td>1.3%</td>
<td>2</td>
<td>gmemt (LUBb: prepost.f, 1)</td>
</tr>
<tr>
<td>[6]</td>
<td>0.060</td>
<td>1.3%</td>
<td>100.0%</td>
<td>0.060</td>
<td>1.3%</td>
<td>2</td>
<td>bdiv (LUBb: LUBb.f, 85)</td>
</tr>
<tr>
<td>[1]</td>
<td>0.000</td>
<td>0.0%</td>
<td>100.0%</td>
<td>4.620</td>
<td>100.0%</td>
<td>154</td>
<td>__start (LUBb: crt1text.s, 103)</td>
</tr>
<tr>
<td>[2]</td>
<td>0.000</td>
<td>0.0%</td>
<td>100.0%</td>
<td>4.620</td>
<td>100.0%</td>
<td>154</td>
<td>main (libf compil: main.c, 76)</td>
</tr>
<tr>
<td>[3]</td>
<td>0.000</td>
<td>0.0%</td>
<td>100.0%</td>
<td>4.620</td>
<td>100.0%</td>
<td>154</td>
<td>LUBb (LUBb: LUBb.f, 1)</td>
</tr>
</tbody>
</table>
### Profiling @ SGI: prof -b

**Butterfly function list, in descending order by inclusive time**

<table>
<thead>
<tr>
<th>attrib.%</th>
<th>attrib.time</th>
<th>incl.time</th>
<th>caller (callsite) [index]</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>4.620</td>
<td>0.0%</td>
<td>__start [1]</td>
</tr>
<tr>
<td>100.0%</td>
<td>4.620</td>
<td>0.0%</td>
<td>main (0x10010128: LUBb: crt1text.s, 177) [2]</td>
</tr>
<tr>
<td>0.0%</td>
<td>0.000</td>
<td>4.620</td>
<td>LUBb (0x0c45f980: libftn.so: main.c, 97) [3]</td>
</tr>
<tr>
<td>4.620</td>
<td></td>
<td></td>
<td>__start (0x10010128: LUBb: crt1text.s, 177) [1]</td>
</tr>
</tbody>
</table>

**Time when called from**

<table>
<thead>
<tr>
<th>attrib.%</th>
<th>attrib.time</th>
<th>incl.time</th>
<th>callee (callsite) [index]</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>4.620</td>
<td>0.0%</td>
<td>main (0x10010128: LUBb: crt1text.s, 177) [2]</td>
</tr>
<tr>
<td>100.0%</td>
<td>4.620</td>
<td>0.0%</td>
<td>LUBb (0x0c45f980: libftn.so: main.c, 97) [3]</td>
</tr>
<tr>
<td>0.0%</td>
<td>0.000</td>
<td>4.620</td>
<td>LUBb (0x10010128: LUBb: crt1text.s, 177) [1]</td>
</tr>
</tbody>
</table>

**Line list, in descending order by function-time and then line number**

<table>
<thead>
<tr>
<th>secs</th>
<th>%</th>
<th>cum. %</th>
<th>samples</th>
<th>function (dso: file, line)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.050</td>
<td>1.0</td>
<td>1.0</td>
<td>5</td>
<td>bmod (LUBb: LUBb.f, 122)</td>
</tr>
<tr>
<td>0.070</td>
<td>1.4</td>
<td>2.3</td>
<td>7</td>
<td>bmod (LUBb: LUBb.f, 143)</td>
</tr>
<tr>
<td>0.050</td>
<td>1.0</td>
<td>3.3</td>
<td>5</td>
<td>bmod (LUBb: LUBb.f, 144)</td>
</tr>
<tr>
<td>0.360</td>
<td>7.0</td>
<td>10.3</td>
<td>36</td>
<td>bmod (LUBb: LUBb.f, 145)</td>
</tr>
<tr>
<td>4.330</td>
<td>83.8</td>
<td>94.0</td>
<td>433</td>
<td>bmod (LUBb: LUBb.f, 146)</td>
</tr>
<tr>
<td>0.060</td>
<td>1.2</td>
<td>95.2</td>
<td>6</td>
<td>bmod (LUBb: LUBb.f, 151)</td>
</tr>
<tr>
<td>0.010</td>
<td>0.2</td>
<td>95.4</td>
<td>1</td>
<td>genmat (LUBb: prepost.f, 10)</td>
</tr>
<tr>
<td>0.090</td>
<td>1.7</td>
<td>97.1</td>
<td>9</td>
<td>genmat (LUBb: prepost.f, 12)</td>
</tr>
<tr>
<td>0.010</td>
<td>0.2</td>
<td>97.3</td>
<td>1</td>
<td>bmod (LUBb: LUBb.f, 104)</td>
</tr>
<tr>
<td>0.050</td>
<td>1.0</td>
<td>98.3</td>
<td>5</td>
<td>bmod (LUBb: LUBb.f, 106)</td>
</tr>
<tr>
<td>0.040</td>
<td>0.8</td>
<td>99.0</td>
<td>4</td>
<td>bmod (LUBb: LUBb.f, 108)</td>
</tr>
<tr>
<td>0.040</td>
<td>0.8</td>
<td>99.8</td>
<td>4</td>
<td>fwd (LUBb: LUBb.f, 183)</td>
</tr>
<tr>
<td>0.010</td>
<td>0.2</td>
<td>100.0</td>
<td>1</td>
<td>LUBb (LUBb: LUBb.f, 35)</td>
</tr>
</tbody>
</table>
Line list, in descending order by function-time and then line number

counts % cum.% samples function (dso: file, line)
16424 0.1 0.1 8 bmod (LUBb: LUBb.f, 122)
2053 0.0 0.1 1 bmod (LUBb: LUBb.f, 143)
4159 0.0 0.1 3 bmod (LUBb: LUBb.f, 144)
26689 0.1 0.3 13 bmod (LUBb: LUBb.f, 145)
17937061 97.3 97.6 8737 bmod (LUBb: LUBb.f, 146)
8212 0.0 97.6 4 bmod (LUBb: LUBb.f, 151)

Higher percentage of L1 misses than time

$ perfex -a -y ./Gauss_seidel
WARNING: Multiplexing events to project totals--inaccuracy possible
iterations: 43
time to compute = 2.739801
Summary for execution of ./Gauss_seidel

Jesus Labarta, MP-2002
### Statistics

- Graduated instructions/cycle: \(0.775664\)
- Graduated floating point instructions/cycle: \(0.515293\)
- Graduated loads & stores/cycle: \(0.209922\)
- Graduated loads & stores/floating point instruction: \(0.407383\)
- Mispredicted branches/Decoded branches: \(0.004188\)
- Graduated loads/Issued loads: \(0.998740\)
- Graduated stores/Issued stores: \(0.999798\)
- Data mispredict/Data cache hits: \(0.053969\)
- Instruction mispredict/Instruction cache hits: \(1.472222\)
- Cache Line Reuse: \(10.372722\)
- Data Cache Hit Rate: \(0.912070\)
- Data Cache Hit Rate: \(0.813321\)
- Time accessing memory/Total time: \(0.641474\)
- Time not making progress (probably waiting on memory) / Total time: \(0.533619\)
- L1-L2 bandwidth used (MB/s, average per process): \(287.557799\)
- Memory bandwidth used (MB/s, average per process): \(215.469597\)
- MFLOPS (average per process): \(128.83352\)

### Event Counter Information

<table>
<thead>
<tr>
<th>Event Counter Name</th>
<th>Counter Value</th>
<th>Type</th>
<th>Time (s)</th>
<th>Min Time (s)</th>
<th>Max Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Cycles ...............................................</td>
<td>6686535352</td>
<td>2.674614</td>
<td>2.674614</td>
<td>2.674614</td>
<td></td>
</tr>
<tr>
<td>16 Cycles ...............................................</td>
<td>6686535352</td>
<td>2.674614</td>
<td>2.674614</td>
<td>2.674614</td>
<td></td>
</tr>
<tr>
<td>21 Graduated floating point instructions........</td>
<td>344553766</td>
<td>1.378211</td>
<td>0.689106</td>
<td>71.666667</td>
<td></td>
</tr>
<tr>
<td>14 ALU/FPU progress cycles .......................</td>
<td>311847136</td>
<td>1.247389</td>
<td>1.247389</td>
<td>1.247389</td>
<td></td>
</tr>
<tr>
<td>7 Quadwords written back from scache...............</td>
<td>17389298</td>
<td>0.465157</td>
<td>0.294221</td>
<td>0.445157</td>
<td></td>
</tr>
<tr>
<td>25 Primary data cache misses .....................</td>
<td>33462420</td>
<td>0.446141</td>
<td>0.139220</td>
<td>0.446141</td>
<td></td>
</tr>
<tr>
<td>2 Issued loads .........................................</td>
<td>57995856</td>
<td>0.383183</td>
<td>0.383183</td>
<td>0.383183</td>
<td></td>
</tr>
<tr>
<td>18 Graduated loads ....................................</td>
<td>56755120</td>
<td>0.382700</td>
<td>0.382700</td>
<td>0.382700</td>
<td></td>
</tr>
<tr>
<td>22 Quadwords written back from primary data cache.</td>
<td>23384556</td>
<td>0.360124</td>
<td>0.235711</td>
<td>0.416247</td>
<td></td>
</tr>
<tr>
<td>3 Issued stores ........................................</td>
<td>44699744</td>
<td>0.178795</td>
<td>0.178795</td>
<td>0.178795</td>
<td></td>
</tr>
<tr>
<td>19 Graduated stores ....................................</td>
<td>2204880</td>
<td>0.088340</td>
<td>0.088340</td>
<td>0.088340</td>
<td></td>
</tr>
<tr>
<td>6 Decoded branches ....................................</td>
<td>22576</td>
<td>0.056149</td>
<td>0.056149</td>
<td>0.056149</td>
<td></td>
</tr>
<tr>
<td>23 TLB misses .........................................</td>
<td>22576</td>
<td>0.056149</td>
<td>0.056149</td>
<td>0.056149</td>
<td></td>
</tr>
<tr>
<td>30 Store/prefetch exclusive to clean block in scache</td>
<td>1088672</td>
<td>0.043555</td>
<td>0.043555</td>
<td>0.043555</td>
<td></td>
</tr>
<tr>
<td>10 Secondary instruction cache misses ............</td>
<td>4704</td>
<td>0.001421</td>
<td>0.001421</td>
<td>0.001421</td>
<td></td>
</tr>
<tr>
<td>24 Mispredicted branches ............................</td>
<td>92496</td>
<td>0.000525</td>
<td>0.000000</td>
<td>0.001931</td>
<td></td>
</tr>
<tr>
<td>9 Primary instruction cache misses ...............</td>
<td>5280</td>
<td>0.000381</td>
<td>0.000119</td>
<td>0.000381</td>
<td></td>
</tr>
<tr>
<td>31 Store/prefetch exclusive to shared block in scache</td>
<td>26288</td>
<td>0.001050</td>
<td>0.001050</td>
<td>0.001050</td>
<td></td>
</tr>
<tr>
<td>1 Issued instructions ................................</td>
<td>504611952</td>
<td>0.000000</td>
<td>0.000000</td>
<td>2.056584</td>
<td></td>
</tr>
<tr>
<td>4 Issued store conditionals ........................</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>5 Failed store conditionals ......................</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>8 Correctable scache data array ECC errors ......</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>11 Instruction misprediction from scache way prediction table</td>
<td>849</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>12 External interventions ...........................</td>
<td>1552</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>13 External invalidations ...........................</td>
<td>13976</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>15 Graduated instructions ...........................</td>
<td>518654196</td>
<td>0.000000</td>
<td>0.000000</td>
<td>2.074338</td>
<td></td>
</tr>
<tr>
<td>17 Graduated instructions ...........................</td>
<td>64050472</td>
<td>0.000000</td>
<td>0.000000</td>
<td>2.026419</td>
<td></td>
</tr>
<tr>
<td>20 Graduated store conditionals ...................</td>
<td>0</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>27 Data misprediction from scache way prediction table</td>
<td>540416</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>28 External intervention hits in scache ..........</td>
<td>1552</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td>29 External invalidation hits in scache ..........</td>
<td>1744</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
</tbody>
</table>
What can we say about an unknown application/system without looking at the source code in short time?
“A single instrumented run captures a lot of information that is essentially thrown away in current parallel programming practice.”

**Philosophy: Flexibility, insight !!!**
- Performance analysis ≡ search on a huge and fuzzy space
  - Be equipped with flexible tools ...
    - No semantics in the tool
  - ...supporting quantitative analysis ...
  - and be ready for surprises

**Core tools**
- Paraver
- Dimemas

... available through [http://www.cepba.upc.es/tools](http://www.cepba.upc.es/tools)
Index

- CEPBA-Tools
- Paraver
  - Description
  - Instrumentation
  - Understanding applications

- Dimemas
  - Description
  - Validation
  - Understanding applications

CEPBA-Tools

- Tracing tools: MPItrace, OMPtrace, OMPitrace, Java
- Translators
  - .ute ➔ UTE2paraver ➔ .cfg
- System monitoring
  - /proc ➔ SCPUs ➔ .prv ➔ .pcf
- Performance simulator
  - .trf ➔ Dimemas
- OpenMP environment: NANOS
  - ➔ NanosCompiler
Paraver: Performance Data browser

Raw data

Performance index: $s(t)$ (piecewise constant)
- Identifier of function
- Hardware counts
- Miss ratios
- Performance (IPC, Mflops, …)
- Routine duration

Tunable

Statistics
- Average miss ratio per routine
- Histogram of routine duration
- Number of messages

Seeing is believing

measuring is better

Visualization

Encoding
- Discrete colors
- Function
- Gradient color
- Not null gradient color

Navigation
- Zoom/Undo
- Y-scale
- Multiple windows
  - synchronize
- Time measurement
- Multiple traces
Visualization: MPI

- Typical state display for MPI programs

- Running state (1)
- Communication (13)
- Group
- P2P communication
- Immediate Send (10)
- Wait (6)
- User events

Visualization: OpenMP

- Running state (1)
- Fork/join state (7)
- Fork/join state (7)
- Idle state (0)
- Entry events
- Exit events

Jesus Labarta, MP-2002
Visualization

- **L1 misses/ms**

- **TLB misses/ms**

- **L2 misses/ms**

Analysis modules: 1D

- **Example measures**
  - Average processor utilization
  - Average duration/variance of specific function (if within range)
  - Number of calls to specific function
  - Number of communications
  - Total cache misses in an interval
  - ...

 Jesús Labarta, MP-2002
Analysis modules: 2D

- 1 column per control window value
  - User Function

- Per thread statistics
  - # times function is called
  - Per function execution profile
  - average function call duration

---

Analysis modules: 2D

- 1 column per control window value
  - Range of Durations

- Per thread statistics
  - Histogram of durations
  - Cumulative time at each duration
Analysis modules: 2D

- Statistics on data window accumulate on column specified by control window
- Per thread statistics
  - Average of data window
  - Integral of data window
  - #data window value changes
  - …

Example measures
- Histogram of function duration
- Correlation between Cache misses and IPC
- Correlation between function duration and hardware counts
Complex?

- **Window configuration files: Capture views**
  - Expert knowledge on how to compute performance index
  - Knowledge on “reasonable” values (scales)
  - Specific time and scales to expose behavior

- **What is useful for**
  - Performance analysis by non expert
  - Training
  - Checkpointing of studies
  - Cooperative work
  - Bug reporting

Configuration files: Directory tree

- **OMPtrace**
  - General
    - State as is, user functions, user function distribution
  - OpenMP
    - Parallel functions, parallel function distribution,
  - MPI
    - MPI call profile, MPI call distribution, message size, send BW, …
  - Counters
    - Program
      - Memory ops mix, Memory access direction, …
    - Architecture
      - L2 miss ratio, …
    - Performance
      - IPC, cycles per ms, MIPS, Memory BW, processor BW, …
Methodology

- Where to look?
  - Potentially very complex
  - Intricate relations between huge number of factors
  - Microscopic phenomena may have macroscopic effects
  - Delayed effects

  ![Diagram showing different units of measurement]

- What to look for?
- How?

Methodology

- The Optimization process

  ![Graph showing optimization process with Obsession and easy to get trapped points]
Methodology: a wish

- A set of performance indices to look at
- Expected observations. Good and bad references
- A mechanism to drive the search process based on precious observations
  - Sequence
  - tree
- Conclusion: exit path when performance problem identified

Reality: still a wish

Methodology and Paraver

- Paraver is not a methodology
- Paraver and the configuration files support the development of methodologies appropriate for families of applications or environments
Methodology: Reference

- What to trace? For how long?
- What is the application structure?

Objective
- Representative duration
  - Relevant part: not initialization
  - Sufficient periods
- Representative events
  - Routines
  - Variables
- Small trace

Analysis methodology

- Scalability
- Parallelized
- Instruction count
- Load balance efficiency
- Processor efficiency
- Synchronization contention
- Cache misses
- Invalidations
- Bandwidth
Instrumentation

Trace format

Type of record: CPU: thread: Record specific information

Application: task: thread

State
Event
Relation

time_start, time_end, state
time, type, value
src, dst, time_src, time_dst, tag, size

Timeline
Trace records (.prv)

#Paraver (07/11/02 at 14:08):418619736199_ns:01:2(8:0,8:0),3
C:1:0:2:1:2
C:1:1:1:1
C:1:2:2:1:2
2:0:1:11:0:40000001:1
1:0:1:11:0:3839439:1
...
1:0:1:1:1:3839439:37684493:15
2:0:1:1:1:11779626:40000001:1
1:0:1:1:1:11779626:15502599:1
1:0:1:1:1:1377926919:378291200:3
1:0:1:1:1:3779318400:37829119:9
1:0:1:1:1:378219119:385980426:1
...

Symbolic information (.pcf)

DEFAULT_OPTIONS
LEVEL THREAD
UNITS NANOSEC
LOOK_BACK 100
SPEED 1
FLAG_ICONS ENABLED
NUM_OF_STATE_COLORS 129
YMAX_SCALE 128

DEFAULT_SEMANTIC THREAD_FUNC State As Is

STATES
0  Idle
1  Running
2  Not created
3  Waiting a message
4  Blocked
5  Thd. Synchr.
6  Wait/WaitAll
7  Sched. and Fork/Join
8  Test/Probe
9  blocking Send
10 Immediate Send
11 Immediate Receive
12 I/O
13 Global OP
14 Tracing Disabled

EVENT_TYPE
0  Event
1  End
28 __areamod_MOD_areaave@OL@2
29 __areamod_MOD_areaave@OL@1
30 __atm_lndmod_MOD_atmlnd_drv@OL@2
31 __atm_lndmod_MOD_atmlnd_drv@OL@1
32 camice@OL@1

EVENT_TYPE
0  60000018 Parallel Function
VALUES
0  End
28 __areamod_MOD_areaave@OL@2
29 __areamod_MOD_areaave@OL@1
30 __atm_lndmod_MOD_atmlnd_drv@OL@2
31 __atm_lndmod_MOD_atmlnd_drv@OL@1
32 camice@OL@1

EVENT_TYPE
0  50000001 MPI Point-to-point
VALUES
1  MPI_Send
2  MPI_Recv
3  MPI_Isend
4  MPI_Irecv
6  MPI_Waitall
41 MPI_Sendrecv
0  End

EVENT_TYPE
0  42000003 Instructions dispatched
0  42000205 Load miss occurred in L1
0  42000512 Processor cycles
0  42000710 Load instr dispatched
Instrumentation

- **Probes**
  - Timing
  - Hardware counters
    - Portable: PAPI (http://icl.utk.edu/projects/papi)
    - Vendor specific: PMAPI, ...

- **Insertion of probes**
  - Dynamic
    - Dltools (http://personals.ac.upc.es/alberts/fpc.html)
    - DPCL (http://oss.software.ibm.com/developerworks/opensource/dpcl)
      - IBM, Linux?
      - Dyninst based (http://www.dyninst.org/)
  - Static:
    - PMPI

OpenMP compilation and Run Time

Source program

```
Call A
A() {
  !omp parallel do
  !omp do i=1,N
  loop body
  enddo
}
```

Compiler generated

```
A() {
xlf_DoInPar
  do i=start,end
  loop body
  enddo
Idle() {
  Compiler generated
```

libomp

```
A@0L1 {
  do i=start,end
  loop body
  enddo
```

Compiler generated
OpenMP instrumentation points

Main thread

- Call A
- A() {
- xlf_DoInPar
- do l=start,end loop body
- enddo

Timeline

1
USR_FCT, idA
HWC, Delta

2
OMP_PAR,1

3
USR_FCT, idA@0L1
HWC, Delta

4
USR_FCT, 0
HWC, Delta

5
OMP_PAR,0

6
USR_FCT, 0
HWC, Delta

Timeline

Slave threads

- Idle_loop() {
- do l=start,end loop body
- enddo

Timeline

0 (idle)

1

Jesus Labarta, MP-2002
Linux Instrumentation : MPI

- Using PMPI interface: Events at entry/exit of MPI calls
- PAPI: include hardware counter events
- MPITrace API: include user callable routines to
  - emit events
    - mpirace_event(int type, int value)
    - mpirace_eventandcounters(int type, int value)
    - mpirace_counters()
  - stop/resume tracing
    - mpirace_shutdown(), mpirace_resume()

Linux tracing

- Use
  setenv MPTRACE_COUNTERS <list_of_hwc_events>
  mpirace mpirun -np <nb procs> -machinefile <hosts> myprogram


**Overhead**

- **Application elapsed time w/o instrumentation**
  - Similar → user happy
  - Very different → the application has a problem
  - Very different → still very useful
    - i.e.: Hardware counts

- **Learn how to live with it**
  - Don’t relax try to extract as much information as possible

- **Overhead of tracing in Linux**
  - No hardware counters: 1-2 µs
  - Hardware counters:
    - 1 Process: 6-7 µs
    - 4 Processes SMP: 16-19 µs (PAPI)

---

**Internals**
**Structure**

- Tracefile
  - Filter
    - Reduced Tracefile: subset of real trace
  - Semantics
    - Function of time (semantic value)
      - Events
        - Representation
          - Visualization
          - Textual
          - Analysis

**Semantic value**

- $S = f(t)$
  - Piecewise constant function of time represented by one window
  - Depends on the filtered subtrace: subset of records of the trace left through by the filter. Each window may see a different subtrace.
  - The semantic value at time $t$ may depend on records with time stamps potentially very far apart from $t$. 
Filter module

- **What:** restrict records that pass to the semantic module
  - Events
    - by type
    - by value
  - Communications
    - by tag
    - by size
    - by source / destination
    - logical / physical
- **What for:**
  - Reduce amount of information to display
  - Feed properly the semantic module

Jesus Labarta, MP 2002
Semantic module

- **Angle:**
  - Process model
    - Thread, task, application, workload
  - Resource model
    - CPU, node, system

**Process view**

**Resource view**

---

Semantic module: Thread function

**Thread function: State as is**

Useful for
- Global thread activity: computing, idle, fork/join, waiting,.....
### Filter: type == 2

**Thread function: Last event value**

- Usefull for:
  - In parallel region
  - Mutual exclusion
  - Variable values: iteration,....

### Filter: type == 4

**Thread function: Next event value**

- Usefull for:
  - Hwc events (TLB, L1 misses,...) within interval
Semantic module: Thread function

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

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

Semantic module: Thread function

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

Useful for
- Routine
Semantic module: process model view

- Semantic value: f(t)
- \( f = f_{\text{comp2}} \circ f_{\text{comp1}} \circ f_{\text{Application}} \circ f_{\text{task}} \circ f_{\text{thread}} \)
- Semantic functions
  - \( f_{\text{comp2}}, f_{\text{comp1}} \): sign, mod, div, in range, select range
  - \( f_{\text{Application}} \): add, average, max, select
  - \( f_{\text{task}} \): add, average, max, select
  - \( f_{\text{thread}} \): in state, useful, given state, last event value, next event value, average next event value, interval between events, ...

Semantic module: resource view

- \( f_{\text{resource}} = f_{\text{System}} \circ f_{\text{Node}} \circ f_{\text{CPU}} \circ f_{\text{thread}} \)
- Semantic functions
  - \( f_{\text{System}} \): add, average, max, select
  - \( f_{\text{Node}} \): add, average, max, select
  - \( f_{\text{CPU}} \): select
  - \( f_{\text{thread}} \): in state, useful, given state, next event value, thread_id
Semantic module

- Derived windows
  - Point wise operation
    - $f = \alpha \cdot f_1 \text{<op>} \beta \cdot f_2$
    - <op>: +, -, *, /, ...

  - L2 Line Loads
  - Mem Ops
  - Loads
  - Stores
  - L2 miss ratio

- Interval between MPI events
- In MPI call
- MPI call duration
Semantic module: process model view

Semantic module: derived windows

- How to build expression

- Multiplying factor
2D analysis

- View control window
  - Analyzed area
- View data window
- Select data window
- Perform an analysis
  - On same area
    - On whole trace (same CW)
    - On whole CW
  - Selecting new CW
- Select statistic

2D analysis

- Control window bin definition
  - Min/max/delta
  - Fit to generate \(\leq 20\) columns
- Statistic color encoding
  - Min/max
  - Fit Min/max value
2D analysis module

- Region analyzed
- Whole table / cell text
- Translate (.pcf)
- Transpose
- Color /not cells
- Hide null columns
- Show/hide lower panel
- Text for cursor
- Bin definition
  - Fit bin size
    - To cover the whole control window dynamic range and generate at most 20 columns
- Color encoding
- Fit color encoding
  - Min and max to dynamic range of statistic

2D analysis

- Right button menu
  - Create a new analyzer window
  - Copy / paste scale:
    - analyzed area (from-to time)
    - Colors
    - Order of columns
  - Change name
  - Save to text
  - Save configuration
Configuration files

- Select trace to which to apply
- Select directory
- List of configuration files in current directory
- Navigate through directory tree
- Description of view of select file

Some examples
Effect of domain partition (sweep3d)
- 1 x 12
- 3 x 4
- 12 x 1

MPI calls analysis
- # MPI calls
- Average MPI call duration
- Percentage of time in MPI call (vs. Total time in MPI calls)
Understanding applications/system

- **Isend duration histogram**
  
  - % Number of calls
  
  - Total time

  ![Isend duration histogram](image)

  - 100-600 µs
  - 1200-3000 µs

- **Isend analysis**
  
  - Duration of I send calls
  
  - Zoom
  
  - Select those with 1200 µs < duration < 3000 µs
  
  - Correlate to general view
    - Longer messages
Dimemas overview

Jesús Labarta, Judit Gimenez
Jordi Caubet, Francesc Escale
CEPBA-UPC

Technology Transfer  Research  Training  Mobility of Researchers
User Support  Education  HPC Facilities  Parallel Expertise

Dimemas

Simulation  Visualization and analysis
Tracing facilities  Message Passing Code
MP library  Parameters modification
Parallel machine  Simulation
Sequential machine  Visualization File
Trace File

Message Passing Code
Visualization and analysis
Parameters modification
Simulation

Jesús Labarta, MP-2002
Characterises application
- Sequence of resource demands for each task
- Sequence of events: communication

Application model

Format
- SDDF for historical reasons
- Definition of records

```
#1: "CPU burst" {
    int "taskid";
    int "thid";
    double "time";
};;

#2: "NX send" {
    int "taskid";
    int "thid";
    int "dest taskid";
    int "msg length";
    int "tag";
    int "commid";
    int "use_rendezvous";
};;

#40: "block begin" {
    int "taskid";
    int "thid";
    int "blockid";
};;

#41: "block end" {
    int "taskid";
    int "thid";
    int "blockid";
};;

#201: "global OP" {
    int "rank";
    int "thid";
    int "glop_id";
    int "comm_id";
    int "root_rank";
    int "root_thid";
    int "bytes_sent";
    int "bytes_recv";
};;
```
Tracefile

- Format
  - ASCII records

```
... 
"block begin" { 35, 0, 73 };;
"NX recv" { 35, 0, 39, 4160, 10003, 0, 1 };;
"block end" { 35, 0, 73 };;
"CPU burst" { 35, 0, 0.000004 };;
"block begin" { 35, 0, 73 };;
"NX recv" { 35, 0, 31, 4160, 10004, 0, 1 };;
"block end" { 35, 0, 73 };;
"CPU burst" { 35, 0, 0.000247 };;
"block begin" { 35, 0, 75 };;
"NX send" { 35, 0, 34, 1560, 10001, 0, 0 };;
"block end" { 35, 0, 75 };;
"CPU burst" { 35, 0, 0.000087 };;
"block begin" { 35, 0, 75 };;
"NX send" { 35, 0, 31, 3460, 10003, 0, 0 };;
"block end" { 35, 0, 75 };;
...
```

Tracefile

- Format
  - ASCII records

```
...
"CPU burst" { 2, 0, 0.006544 };;
"block begin" { 2, 0, 4 };;
"global OP" { 2, 0, 8, 0, 0, 0, 21024, 21024 };;
"block end" { 2, 0, 4 };;
"CPU burst" { 2, 0, 0.064627 };;
"block begin" { 2, 0, 4 };;
"global OP" { 2, 0, 8, 0, 0, 0, 21024, 21024 };;
"block end" { 2, 0, 4 };;
"CPU burst" { 2, 0, 0.008348 };;
...
```
Instrumentation

- Dimemas instrumentation
  - MPI\texttt{D}trace
    - Run the same way as OMPI\texttt{trace}

- Tracefile generated by OMPI\texttt{trace}
  - Paraver: \texttt{Trace Generation}
    - The Paraver trace should have been obtained with dedicated resources
    - Does not support selection of specific part of the trace
    - Possible to get rid of non relevant events

Parallel machine model

- Network of SMPs
  - Multiprogrammed workload

- Key factors influencing performance
  - "Abstract" architecture
  - Basic MPI protocols
  - No attempt to model details of a specific implementation

- Objectives
  - Simple/general
  - Fast simulation
Latency
- Depends on node. Independent of message size
- Paid by sends and receives at beginning of operation
- Uses CPU

Network Bandwidth
- Independent of message size
- Uses links and buses
  - Links:
    - determine concurrent accesses to the network
    - Half / Full Duplex
  - Busses:
    - maximum number of simultaneous transfers
  - Output link, input link and bus must be allocated to start transmission

\[ T = \text{Latency} + \frac{\text{msg size}}{\text{Bandwidth}} \]
**p2p communication model**

- **Late receiver**
  - Machine Latency
  - Uses CPU
  - Independent of size
  - Simulated contention for machine resources (links & buses)

  ![Late receiver diagram](image)

- **Rendezvous**
  - Machine Latency
  - Uses CPU
  - Independent of size
  - Process Blocked

  ![Rendezvous diagram](image)
Collective communication model

- Phases
  - Barrier
  - Fan-in
  - Fan-out

- Cost of communication phase
  - Generic
  - Per call

Collective communication model

- Generic model
  - Barrier / Fan-in / Fan-out
  - Cost of communication phase
    - Generic
  - Per call
    - Model factor
      - Lin / log / const
    - Size of message
      - Min over all processes
      - Avg over all processes
      - Max over all processes
Collective Communication Model

**Generic model**
- Communication time

\[
\text{Time} = \left( \text{Latency} + \frac{\text{Size}}{\text{Bandwidth}} \right) \times \text{MODEL\_FACTOR}
\]

- Model factor
  - Lin / log / const

<table>
<thead>
<tr>
<th>Model</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>0</td>
</tr>
<tr>
<td>Constant</td>
<td>1</td>
</tr>
<tr>
<td>Linear</td>
<td>( P )</td>
</tr>
<tr>
<td>Logarithmic</td>
<td>( \sum_{i=1}^{N\text{steps}} \frac{1}{\text{steps}_i} = \frac{C}{B} )</td>
</tr>
</tbody>
</table>

Jesus Labarta, MP-2002

Collective Communication Model

**Per call model**
- Model factor
  - Lin
  - Log
  - Const

- Size of message
  - Min over all processes
  - Mean over all processes
  - Max over all processes

- Specified in input file

<table>
<thead>
<tr>
<th>Operation</th>
<th>IN Model</th>
<th>IN Size</th>
<th>OUT Model</th>
<th>OUT Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barrier</td>
<td>LIN</td>
<td>MAX</td>
<td>LIN</td>
<td>MAX</td>
</tr>
<tr>
<td>Bcast</td>
<td>LOG</td>
<td>MAX</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>Gather</td>
<td>LOG</td>
<td>MEAN</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>Gatherv</td>
<td>LOG</td>
<td>MEAN</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>Scatter</td>
<td>NULL</td>
<td>LOG</td>
<td>MEAN</td>
<td>MEAN</td>
</tr>
<tr>
<td>Scatters</td>
<td>NULL</td>
<td>NULL</td>
<td>LOG</td>
<td>MEAN</td>
</tr>
<tr>
<td>Allgather</td>
<td>LOG</td>
<td>MEAN</td>
<td>LOG</td>
<td>MEAN</td>
</tr>
<tr>
<td>Allgatherv</td>
<td>LOG</td>
<td>MEAN</td>
<td>LOG</td>
<td>MEAN</td>
</tr>
<tr>
<td>Alltoall</td>
<td>LOG</td>
<td>MEAN</td>
<td>LOG</td>
<td>MAX</td>
</tr>
<tr>
<td>Alltoallv</td>
<td>LOG</td>
<td>MEAN</td>
<td>LOG</td>
<td>MAX</td>
</tr>
<tr>
<td>Reduce</td>
<td>LOG</td>
<td>2MAX</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>Allreduce</td>
<td>LOG</td>
<td>2MAX</td>
<td>LOG</td>
<td>MAX</td>
</tr>
<tr>
<td>Reduce_Scatter</td>
<td>LOG</td>
<td>2MAX</td>
<td>LOG</td>
<td>MIN</td>
</tr>
<tr>
<td>Scan</td>
<td>LOG</td>
<td>MAX</td>
<td>LOG</td>
<td>MAX</td>
</tr>
</tbody>
</table>

Jesus Labarta, MP-2002
Dimemas GRID: model extension

- Dedicated connections
- External network
  - Variation on effective bandwidth due to traffic
- Collective communication extension

Dimemas GRID: communication model

Decomposition of latency in two

Machine Latency
- Machine resources contention (simulated links & buses)

Transfer Size
- Size

Flight (distance)
- WAN contention (traffic)
Collective operation model

Collective operations configuration file

<table>
<thead>
<tr>
<th>Id_op</th>
<th>MODEL_IN</th>
<th>SIZE_IN</th>
<th>MODEL_OUT</th>
<th>SIZE_OUT</th>
<th>Collective operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LIN</td>
<td>MAX</td>
<td>LIN</td>
<td>MAX</td>
<td>/* MPI_Barrier */</td>
</tr>
<tr>
<td>1</td>
<td>LOG</td>
<td>MAX</td>
<td>0</td>
<td>MAX</td>
<td>/* MPI_Bcast */</td>
</tr>
<tr>
<td>2</td>
<td>LOG</td>
<td>MEAN</td>
<td>0</td>
<td>MAX</td>
<td>/* MPI_Gather */</td>
</tr>
<tr>
<td>3</td>
<td>LOG</td>
<td>MEAN</td>
<td>0</td>
<td>MAX</td>
<td>/* MPI_Gatherv */</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>MAX</td>
<td>LOG</td>
<td>MEAN</td>
<td>/* MPI_Scatter */</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>MAX</td>
<td>LOG</td>
<td>MEAN</td>
<td>/* MPI_Scatterv */</td>
</tr>
<tr>
<td>6</td>
<td>LOG</td>
<td>MEAN</td>
<td>LOG</td>
<td>MEAN</td>
<td>/* MPI_Allgather */</td>
</tr>
<tr>
<td>7</td>
<td>LOG</td>
<td>MEAN</td>
<td>LOG</td>
<td>MEAN</td>
<td>/* MPI_Allgatherv */</td>
</tr>
<tr>
<td>8</td>
<td>LOG</td>
<td>MEAN</td>
<td>LOG</td>
<td>MAX</td>
<td>/* MPI_Alltoall */</td>
</tr>
<tr>
<td>9</td>
<td>LOG</td>
<td>MEAN</td>
<td>LOG</td>
<td>MAX</td>
<td>/* MPI_Alltoallv */</td>
</tr>
<tr>
<td>10</td>
<td>LOG</td>
<td>2MAX</td>
<td>0</td>
<td>MAX</td>
<td>/* MPI_Reduce */</td>
</tr>
<tr>
<td>11</td>
<td>LOG</td>
<td>2MAX</td>
<td>LOG</td>
<td>MAX</td>
<td>/* MPI_Allreduce */</td>
</tr>
<tr>
<td>12</td>
<td>LOG</td>
<td>2MAX</td>
<td>LOG</td>
<td>MIN</td>
<td>/* MPI_Reduce_Scatter*/</td>
</tr>
<tr>
<td>13</td>
<td>LOG</td>
<td>MAX</td>
<td>LOG</td>
<td>MAX</td>
<td>/* MPI_Scan */</td>
</tr>
</tbody>
</table>

Collective operations configuration file

Dimemas GRID: other features

- Variable bandwidth
  - Function of time
  - Function of traffic
Architecture description

- Configuration file
  - SDDF format for historical reasons
  - Definition of records

#1:
"environment information" {
  char "machine_name"[];
  int "machine_id";
  // "instrumented_architecture" "Architecture used to instrument"
  char "instrumented_architecture"[];
  // "number_of_nodes" "Number of nodes on virtual machine"
  int "number_of_nodes";
  // "network_bandwidth" "Data transfer rate between nodes in Mbyte/s"
  // "0 means instantaneous communication"
  double "network_bandwidth";
  // "number_of_buses_on_network" "Maximum number of messages on network"
  // "0 means no limit"
  // "1 means bus contention"
  int "number_of_buses_on_network";
  // "1 Constant, 2 Linear, 3 Logarithmic"
  int "communication_group_model";
};

#2:
"node information" {
  int "machine_id";
  // "node_id" "Node number"
  int "node_id";
  // "simulated_architecture" "Architecture node name"
  char "simulated_architecture"[];
  // "number_of_processors" "Number of processors within node"
  int "number_of_processors";
  // "number_of_input_links" "Number of input links in node"
  int "number_of_input_links";
  // "number_of_output_links" "Number of output links in node"
  int "number_of_output_links";
  // "startup_on_local_communication" "Communication startup"
  double "startup_on_local_communication";
  // "startup_on_remote_communication" "Communication startup"
  double "startup_on_remote_communication";
  // "speed_ratio_instrumented_vs_simulated" "Relative processor speed"
  double "speed_ratio_instrumented_vs_simulated";
  // "memory_bandwidth" "Data transfer rate into node in Mbyte/s"
  // "0 means instantaneous communication"
  double "memory_bandwidth";
  double "external_net_startup";
};
Architecture description

- **Configuration**

  - "wide area network information" ["", 4, 1, 4, 0.0, 0.0, 3];
  - "dedicated connection information" (0, 0, 1, 80.0, [3] (0,1,2), 0, "-", "-", 1024, "-", [2] (0,1), 0.0, 0.0,0.0);
  - "environment information" ["", 3, ",", 1, 0.0, 0, 1];
  - "environment information" ["", 2, ",", 1, 0.0, 0, 3];
  - "environment information" ["", 1, ",", 1, 0.0, 0, 2];
  - "environment information" ["", 0, ",", 1, 0.0, 0, 3];
  - "node information" [0, 0, ",", 4, 4, 0.0, 0.0, 1.0, 0.0, 0.0];
  - "node information" [3, 1, ",", 1, 3, 0.0, 0.0, 1.0, 0.0, 0.0];
  - "node information" [2, 2, ",", 1, 3, 0.0, 0.0, 1.0, 0.0, 0.0];
  - "node information" [3, 3, ",", 3, 3, 0.0, 0.0, 1.0, 0.0, 0.0];
  - "mapping information" ["/scratch/traces/cpmd/Dimemas_Demo/mar_chunk.trf", 16, [16] {0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3}];
  - "configuration files" ["", "" /scratch/traces/cpmd/Dimemas_Demo/collectivas.cfg", ""];
  - "modules information" [124, 0.5];

Dimemas GUI

- [Image of Dimemas GUI interface]

Jesus Labarta, MP 2002
The simulator: Dimemas

Qualitative validation

Dedicated machine

Shared environment

+ Dimemas
Stability validation

- On loaded system, 30 times
  - trace & measure elapsed time
  - predict time for dedicated system

![Graph showing LU NPB tasks for different numbers of tasks (16, 32, 8) with time on the x-axis and number of tasks on the y-axis.]

![Graph showing predicted vs. actual time for different numbers of tasks (16, 32, 8) with R^2 = 0.9994.]

NAS Benchmarks

- BT, CF, FFT, MG, IS LU, SP
- Class W and A
- P = 8..32

- Target machine model
  - L = 27, BW = 80, B = ∞
Application Analysis

- Load balanced and dependence problems?
  - $BW = \infty$, $L = 0$

- Group messages?
  - $L = \ldots$, $BW = \infty$

- Bandwidth problem?
  - $BW = \ldots$, $L = 0$

- Concurrent communication problems?
  - $BW = \text{target}$, $L = \text{target}$, buses = 1, 2, ...

Bandwidth Trade-offs

- Injection mechanism
Bandwidth Trade-offs

- $L=50 \mu s$, $BW=8\text{ MB/s}$
- $L=50 \mu s$, $BW=4\text{ MB/s}$
- $L=50 \mu s$, $BW=2\text{ MB/s}$

$\Rightarrow 2\text{ MB/s is a problem ...}$

Bandwidth Trade-offs

- $L=50 \mu s$, $BW=2\text{ MB/s}$ but 8 links to network per node
- $L=50 \mu s$, $BW=4\text{ MB/s}$ but 1 bus

... trade off raw bandwidth - bisection/connection BW
System characterization

< 10% error regions

1 half duplex link !!!

Understanding architectures

Machine A

Dimemas Trace

Machine B

Arch. Parameters

Vampir Trace
Understanding architectures

Machine A

- Dimemas Trace
- Arch. Parameters

Dimemas

- Compute Load balance

Machine B

- Vampir Trace

Understanding applications (MPIRE)

- 32 procs (no network contention)

L=25us, BW=100MB/s

L=1000us, BW=100MB/s

L=25us, BW=10MB/s

All windows same scale
**Understanding applications (MPIRE)**

- **Cluster of SMPs**
  - 4nodesx4, 1 link
  - 128 p, L=25, BW=100, no network contention

**Mapping influence**

- 128 p, L=25, BW=100, 4nodesx4, 1 link, no network contention
Understanding architectures

- Unexpected behavior
  - IFS on E10000 (MPICH)*
  - IBM SP (@KTH)

* Courtesy FECIT

Sensitivity analysis

- How sensitive is my program to bandwidth, latency, injection mechanism, routine optimization?
  - How much do they influence the execution time?
    - ST-ORM: “GRID” tool for specifying studies (montecarlo, optimization, parametric studies…), generating and submitting the jobs, collecting and analyzing results.
    - Is it possible to build simple models of program performance?

- Which parts of my program are more sensitive to bandwidth, latency, routine optimization?
Montecarlo studies

- Contention

![Graph 1: Montecarlo studies](image1)

- MPIRE: 32 CPUs (no network contention)

![Graph 2: MPIRE: 32 CPUs](image2)
Monte Carlo studies

- SCF: 32 CPUs (no network contention)

Essentially Sensitive to raw processor performance

Program section sensitivity

to contention

to bandwidth

to latency
Convolution
- Observed performance is a convolution of
  - algorithm characteristics
    - Instructions
    - Communication demands
  - machine characteristics
    - Processor
    - Communication

Tracing is an attempt to deconvolve the algorithm and machine characteristics from an actual run to later convolve (Metasim/Dimemas) with the hypothetical target machine characteristics.

Cooperation with Allan Snawelly (SDSC)

Estimate CPU ratios between tracing and target machine
- Based on simple instruction level simulation
- Started by simple hypothesis
  - IPC \( \propto \) bandwidth
Dimemas GRID: example

- **URANUS**
  - 1 machine, 8 nodes, 1 proc/node
  - latency=30 msecs, BW=0.8 MB/S
  - 2 machines, 4 nodes, 1 processor/node

Example: Uranus

- 64 processes
- 4 16-way SMPs
- Fligh times
  - 0,1,10 and 50 ms.
Example: linpack

- 256 processes
- 16 16-way SMPs
- BW (MB/s) / Flight time (ms)
  - 50 / 1
  - 100 / 1
  - 200 / 1
  - 200 / 0.1
  - 200 / 0.01
  - 500 / 0.01

Example: Explore response surface

- Linpack
- 256 processes
- 16 16-way SMPs
- Flight time and bandwidth exploration
**RNAfold results**

Machines involved:
- Cray T3Y/900 at HLRS
- IBM SP-3 at CEPBA
- SGI O2000 at CEPBA

Application:
- RNAfold

Configurations
- 4+4 processors
- 6+6 processors
- 14+14 processors

*Performance Prediction in a Grid environment, Rosa M. Badia, Francesc Escalé, Edgar Gabriel, Judit Gimenez, Rainer Keller, Jesús Labarta, Matthias S. Müller, ACROS GRIDS 2003*

---

**Parametric studies (RNAfold)**

Yeast RNA
4+4 processors
http://www.cepba.upc.es/dimemas

cepbatools@cepba.upc.es