Tools

Jesús Labarta

CEPBA-UPC

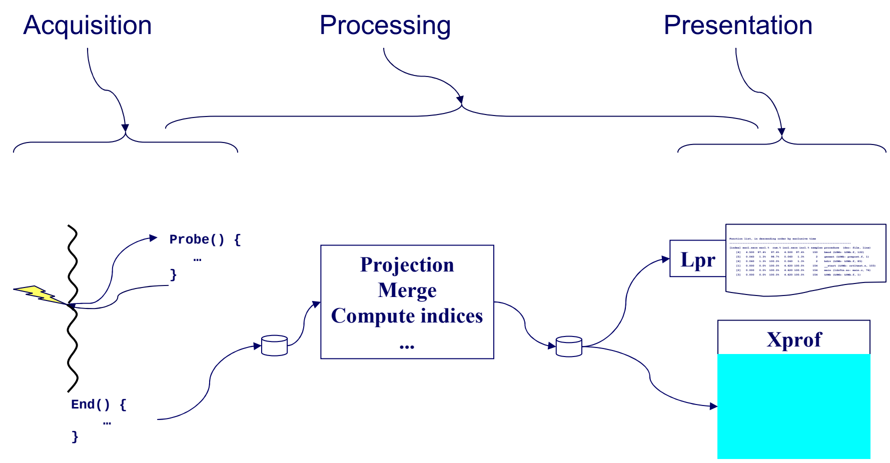
Technology Transfer Research Training Mobility of Researchers
User Support Education HPC Facilities Parallel Expertise

Performance tools

- **Objective:**
  - Identify performance problems and help optimize application
- **Phases**
  - Data acquisition
  - Processing
    - ✓ Compaction
    - ✓ Summarization: Statistics
  - Presentation
    - ✓ Textual
    - ✓ Graphical
    - ✓ Esoteric …

Jesus Labarta, MP,2002

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

Data acquisition

- **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

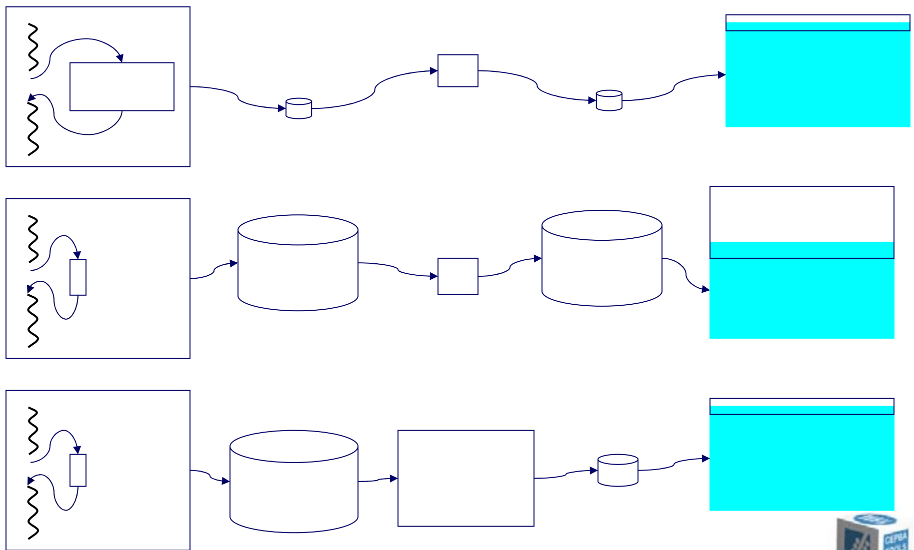
Presentation

- **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



Jesus Labarta, MP,2002

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

Jesus Labarta, MP,2002

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

Profiling @ SGI

```
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
```

Jesus Labarta, MP,2002

Profiling @ SGI

```
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?

Profiling @ SGI: prof

```
Function list, in descending order by exclusive time
-------------------------------------------------------------------------
[index] excl.secs excl.%   cum.% incl.secs incl.% samples procedure  (dso: file, line)
    [4]     4.500  97.4%   97.4%     4.500  97.4%     150 bmod (LUBb: LUBb.f, 122)
    [5]     0.060   1.3%   98.7%     0.060   1.3%       2 genmat (LUBb: prepost.f, 1)
    [6]     0.060   1.3%  100.0%     0.060   1.3%       2 bdiv (LUBb: LUBb.f, 85)
    [1]     0.000   0.0%  100.0%     4.620 100.0%     154 __start (LUBb: crt1text.s, 103)
    [2]     0.000   0.0%  100.0%     4.620 100.0%     154 main (libftn.so: main.c, 76)
    [3]     0.000   0.0%  100.0%     4.620 100.0%     154 LUBb (LUBb: LUBb.f, 1)
```

Routine

Source and line

Just by this routine

Including functions called by this routine

Profiling @ SGI: prof -b

```
Butterfly function list, in descending order by inclusive time
-------------------------------------------------------------------------
        attrib.% attrib.time                         incl.time  caller (callsite) [index]
 [index]   incl.%   incl.time    self%   self-time      procedure [index]
                              attrib.% attrib.time    incl.time  callee (callsite) [index]
-------------------------------------------------------------------------
     [1]   100.0%     4.620     0.0%      0.000        __start [1]
                              100.0%      4.620       4.620  main (@0x10001428; LUBb: crt1text.s, 177) [2]
-------------------------------------------------------------------------
           100.0%     4.620                           4.620  __start (@0x10001428; LUBb: crt1text.s, 177) [1]
     [2]   100.0%     4.620     0.0%      0.000        main [2]
                              100.0%      4.620       4.620  LUBb (@0x0c45f980; libftn.so: main.c, 97) [3]
-------------------------------------------------------------------------
           100.0%     4.620                           4.620  main (@0x0c45f980; libftn.so: main.c, 97) [2]
     [3]   100.0%     4.620     0.0%      0.000        LUBb [3]
                               97.4%      4.500       4.500  bmod (@0x10001608; LUBb: LUBb.f, 35) [4]
                                1.3%      0.060       0.060  bdiv (@0x100015e0; LUBb: LUBb.f, 32) [6]
                                1.3%      0.060       0.060  genmat (@0x100014d8; LUBb: LUBb.f, 16) [5]
-------------------------------------------------------------------------
            97.4%     4.500                           4.620  LUBb (@0x10001608; LUBb: LUBb.f, 35) [3]
     [4]    97.4%     4.500    97.4%      4.500        bmod [4]
-------------------------------------------------------------------------
             1.3%     0.060                           4.620  LUBb (@0x100014d8; LUBb: LUBb.f, 16) [3]
     [5]     1.3%     0.060     1.3%      0.060        genmat [5]
-------------------------------------------------------------------------
             1.3%     0.060                           4.620  LUBb (@0x100015e0; LUBb: LUBb.f, 32) [3]
     [6]     1.3%     0.060     1.3%      0.060        bdiv [6]
```

Time when called from

calls

Jesus Labarta, MP,2002

Profiling @ SGI: prof -l (pcsamp)

```
Line list, in descending order by function-time and then line number
-------------------------------------------------------------------------
          secs      %    cum.%   samples  function (dso: file, line)

         0.050    1.0    1.0         5  bmod (LUBb: LUBb.f, 122)
         0.070    1.4    2.3         7  bmod (LUBb: LUBb.f, 143)
         0.050    1.0    3.3         5  bmod (LUBb: LUBb.f, 144)
         0.360    7.0   10.3        36  bmod (LUBb: LUBb.f, 145)
         4.330   83.8   94.0       433  bmod (LUBb: LUBb.f, 146)
         0.060    1.2   95.2         6  bmod (LUBb: LUBb.f, 151)

         0.010    0.2   95.4         1  genmat (LUBb: prepost.f, 10)
         0.090    1.7   97.1         9  genmat (LUBb: prepost.f, 12)

         0.010    0.2   97.3         1  bdiv (LUBb: LUBb.f, 104)
         0.050    1.0   98.3         5  bdiv (LUBb: LUBb.f, 106)
         0.040    0.8   99.0         4  bdiv (LUBb: LUBb.f, 108)

         0.040    0.8   99.8         4  fwd (LUBb: LUBb.f, 183)

         0.010    0.2  100.0         1  LUBb (LUBb: LUBb.f, 35)
```

Jesus Labarta, MP,2002

Profiling @ SGI: prof -l (dc_hwc)

```
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)
          6159   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)

         22583   0.1   97.7       11  bdiv (LUBb: LUBb.f, 85)
          4106   0.0   97.8        2  bdiv (LUBb: LUBb.f, 104)
          6159   0.0   97.8        3  bdiv (LUBb: LUBb.f, 105)
         67749   0.4   98.2       33  bdiv (LUBb: LUBb.f, 106)
        117021   0.6   98.8       57  bdiv (LUBb: LUBb.f, 108)

          2053   0.0   98.8        1  fwd (LUBb: LUBb.f, 160)
          8212   0.0   98.9        4  fwd (LUBb: LUBb.f, 181)
        188876   1.0   99.9       92  fwd (LUBb: LUBb.f, 183)

          4106   0.0   99.9        2  genmat (LUBb: prepost.f, 11)
          4106   0.0   99.9        2  genmat (LUBb: prepost.f, 12)

          4106   0.0   99.9        2  LUBb (LUBb: LUBb.f, 34)
          2053   0.0  100.0        1  LUBb (LUBb: LUBb.f, 35)

          2053   0.0  100.0        1  lu0 (LUBb: LUBb.f, 70)
          4106   0.0  100.0        2  lu0 (LUBb: LUBb.f, 72)

          2053   0.0  100.0        1  memset (libc.so.1: bzero.s, 160)
```

Higher percentage of L1 misses than time

Profiling @ SGI: perfex –a -y

```
$ 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
```

Profiling @ SGI: perfex –a -y

Event Counter Name	Counter Value	Typ Time (s)	Min Time (s)	Max Time (s)
0 Cycles	668653552	2.674614	2.674614	2.674614
16 Cycles	668653552	2.674614	2.674614	2.674614
21 Graduated floating point instructions	344552768	1.378211	0.689106	71.666976
14 ALU/FPU progress cycles	311847136	1.247389	1.247389	1.247389
26 Secondary data cache misses	2328720	0.703273	0.459782	0.782450
7 Quadwords written back from scache	17388928	0.445157	0.294221	0.445157
25 Primary data cache misses	12342240	0.444814	0.139220	0.444814
2 Issued loads	95795856	0.383183	0.383183	0.383183
18 Graduated loads	95675120	0.382700	0.382700	0.382700
22 Quadwords written back from primary data cache	23384656	0.360124	0.293711	0.416247
3 Issued stores	44698768	0.178795	0.178795	0.178795
19 Graduated stores	44689744	0.178759	0.178759	0.178759
6 Decoded branches	22084880	0.088340	0.088340	0.088340
23 TLB misses	22576	0.006149	0.006149	0.006149
30 Store/prefetch exclusive to clean block in scache	1088672	0.004355	0.004355	0.004355
10 Secondary instruction cache misses	4704	0.001421	0.000929	0.001581
24 Mispredicted branches	92496	0.000525	0.000237	0.001931
9 Primary instruction cache misses	5280	0.000381	0.000119	0.000381
31 Store/prefetch exclusive to shared block in scache	26288	0.000105	0.000105	0.000105
1 Issued instructions	506411952	0.000000	0.000000	2.025648
4 Issued store conditionals	0	0.000000	0.000000	0.000000
5 Failed store conditionals	0	0.000000	0.000000	0.000000
8 Correctable scache data array ECC errors	0	0.000000	0.000000	0.000000
11 Instruction misprediction from scache way prediction table	848	0.000000	0.000000	0.000003
12 External interventions	1552	0.000000	0.000000	0.000000
13 External invalidations	13376	0.000000	0.000000	0.000000
15 Graduated instructions	518634496	0.000000	0.000000	2.074538
17 Graduated instructions	506604672	0.000000	0.000000	2.026419
20 Graduated store conditionals	0	0.000000	0.000000	0.000000
27 Data misprediction from scache way prediction table	540416	0.000000	0.000000	0.002162
28 External intervention hits in scache	1552	0.000000	0.000000	0.000000
29 External invalidation hits in scache	1744	0.000000	0.000000	0.000000

Profiling @ SGI: perfex –a -y

Statistics	
Graduated instructions/cycle	0.775640
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 scache hits	0.053969
Instruction mispredict/Instruction scache hits	1.472222
L1 Cache Line Reuse	10.372722
L2 Cache Line Reuse	4.300010
L1 Data Cache Hit Rate	0.912070
L2 Data Cache Hit Rate	0.811321
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)	215.469957
MFLOPS (average per process)	128.823352

CEPBA Tools

Jesús Labarta

CEPBA-UPC

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

CEPBA tools: Challenge

What can we say

about an unknown application/system

without looking at the source code

in short time?

Jesus Labarta, MP,2002

CEPBA tools: Thesis

"A **single instrumented run**

captures a **lot of information**

that is essentially **thrown away**

in current parallel programming practice"

Jesus Labarta, MP,2002

CEPBA-Tools

- **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

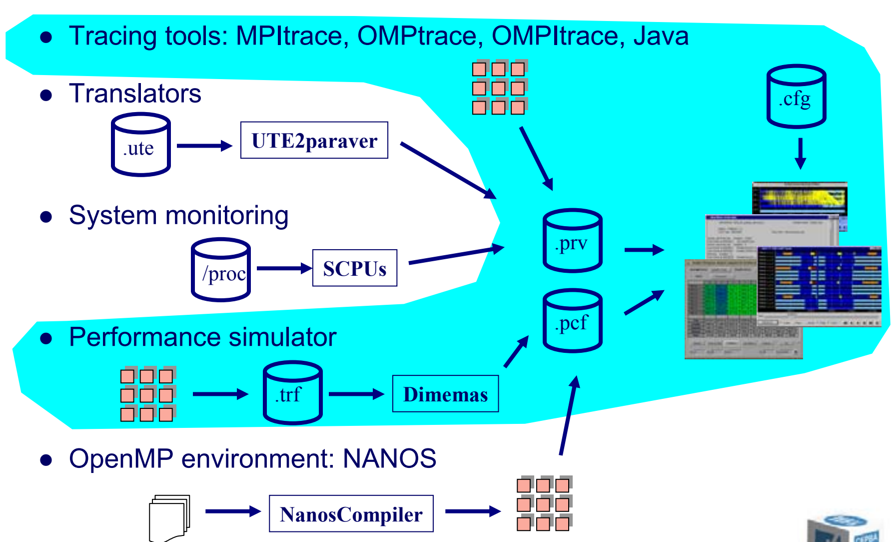
Jesus Labarta, MP,2002

Index

- **CEPBA-Tools**
- **Paraver**
  - Description
  - Instrumentation
  - Understanding applications
- **Dimemas**
  - Description
  - Validation
  - Understanding applications

CEPBA-Tools

- Tracing tools: MPItrace, OMPtrace, OMPItrace, Java
- Translators
- System monitoring
- Performance simulator
- OpenMP environment: NANOS



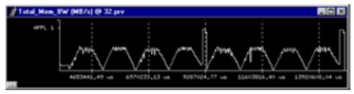
Paraver: Performance Data browser

Raw data

tunable

Performance index : s(t) (piecewise constant)

- Identifier of function
- Hardware counts
- Miss ratios
- Performance (IPC, Mflops,…)
- Routine duration
- …

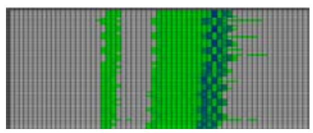


Seeing is believing

measuring is better

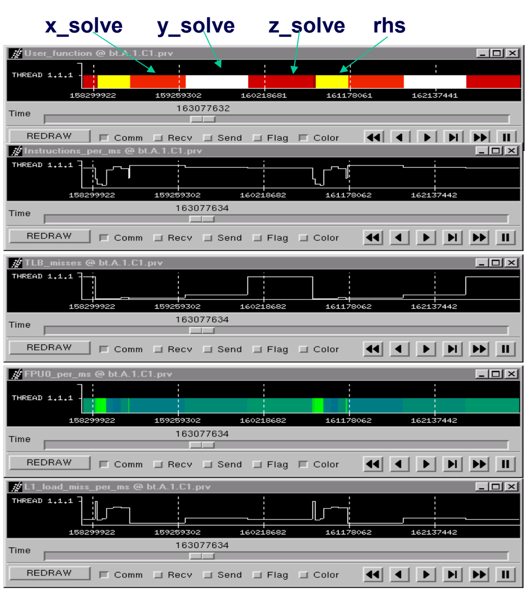
Statistics

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



Jesus Labarta, MP,2002

Visualization



- **Encoding**
  - Discrete colors
  - Function
  - Gradient color
  - Not null gradient color
- **Navigation**
- **Zoom/Undo**
- **Y - scale**
- **Multiple windows**
  - synchronize
- **Time measurement**
- **Multiple traces**

Jesus Labarta, MP,2002

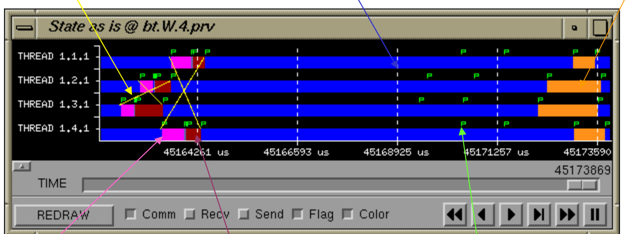
Visualization: MPI

- Typical state display for MPI programs

Group Communication (13)

Running state (1)

P2P communication



Immediate Send (10)

Wait (6)

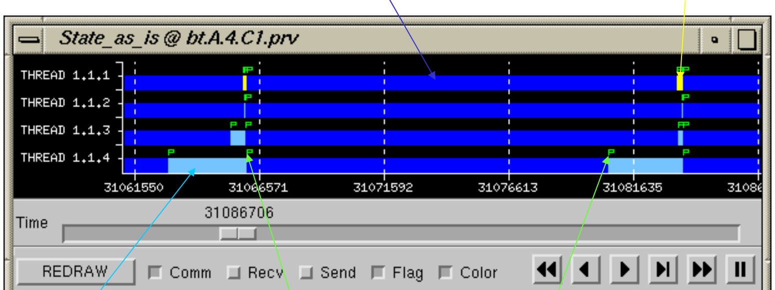
User events

Jesus Labarta, MP,2002

Visualization: OpenMP

Running state (1)

Fork/join state (7)



Idle state (0)

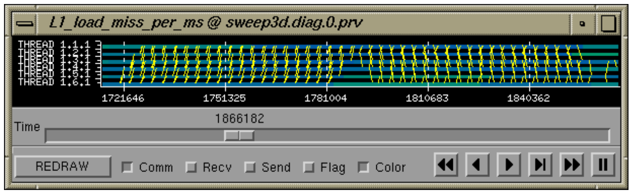
Entry events

Exit events

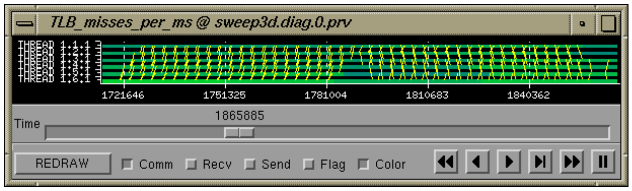
Jesus Labarta, MP,2002

Visualization

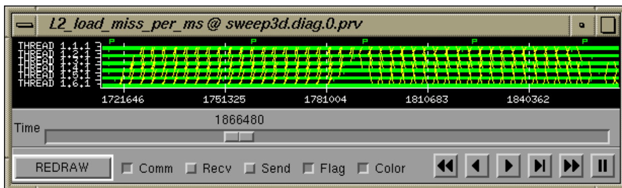
- **L1 misses/ms**



- **TLB misses/ms**

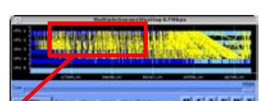


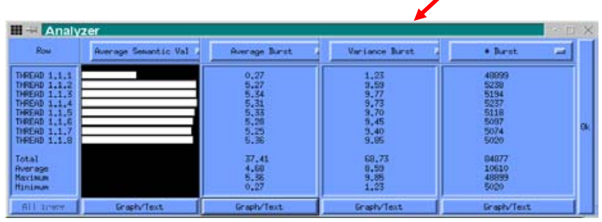
- **L2 misses/ms**



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
  - ...

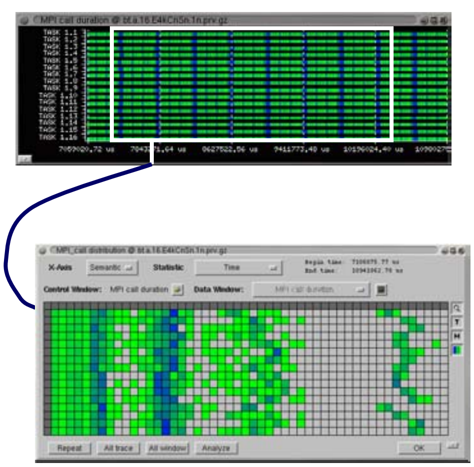




Analysis modules: 2D

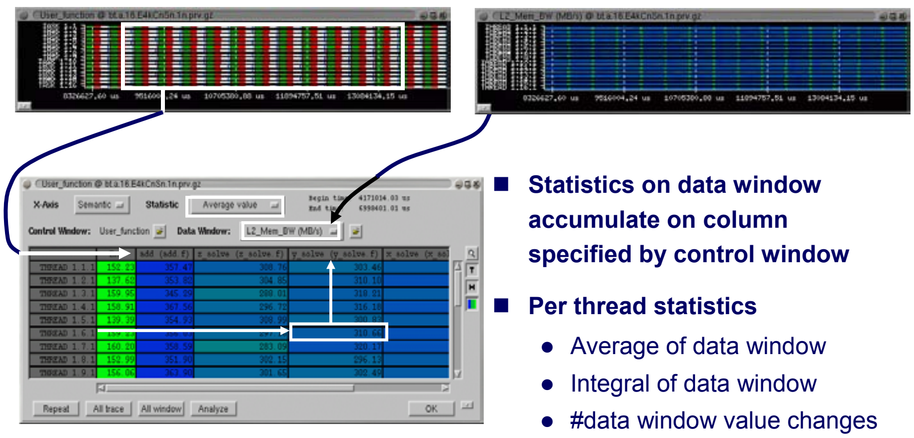
- **1 column per control window value**
  - User Function
- **Per thread statistics**
  - # times function is called
  - Per function execution profile
  - average function call duration

Analysis modules: 2D



- **1 column per control window value**
  - Range of Durations
- **Per thread statistics**
  - Histogram of durations
  - Cumulative time at each duration

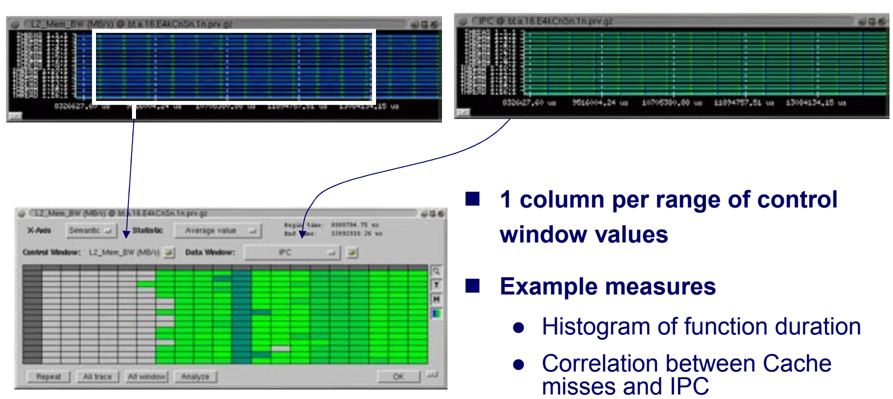
Analysis modules: 2D



- **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
  - …

Jesus Labarta, MP,2002

Analysis modules: 2D



- **1 column per range of control window values**
- **Example measures**
  - Histogram of function duration
  - Correlation between Cache misses and IPC
  - Correlation between function duration and hardware counts

Jesus Labarta, MP,2002

Complex?

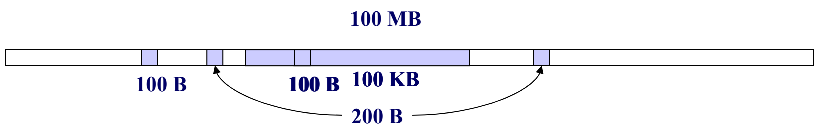
- **Window configuration files: Capture views**
  - Expert knowledge on how to compute performance index
  - Knowledge on “reasonable” values (scales)
  - Specific time and scales to expose behavior
- **What is useful for**
  - Performance analysis by non expert
  - Training
  - Checkpointing of studies
  - Cooperative work
  - Bug reporting

Configuration files: Directory tree

- **OMPItrace**
  - 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

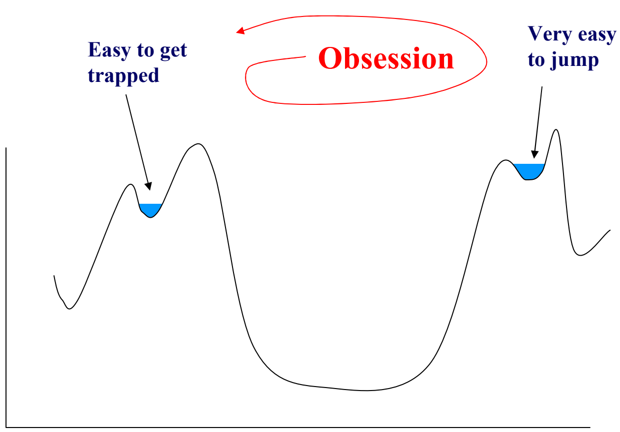
- **Where to look?**
  - Potentially very complex
  - Intricate relations between huge number of factors
  - Microscopic phenomena may have macroscopic effects
  - Delayed effects



- **What to look for?**
- **How?**

Methodology

- **The Optimization process**



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

Jesus Labarta, MP,2002

Methodology and Paraver

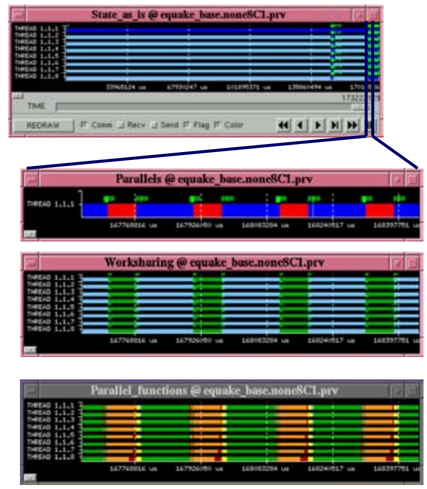
- Paraver is not a methodology
- Paraver and the configuration files support the development of methodologies appropriate for families of applications or environments

Jesus Labarta, MP,2002

Methodology: Reference

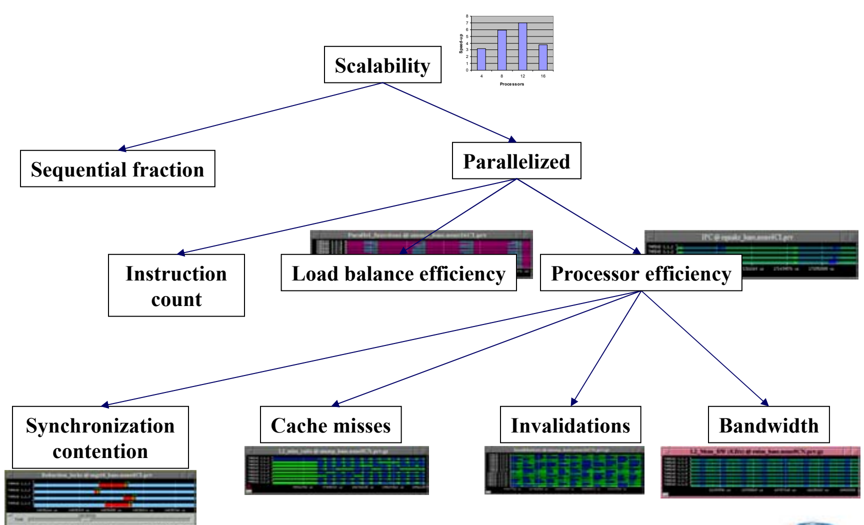
- What to trace? For how long?
- What is the application structure?

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



Jesus Labarta, MP,2002

Analysis methodology



Jesus Labarta, MP,2002

Instrumentation

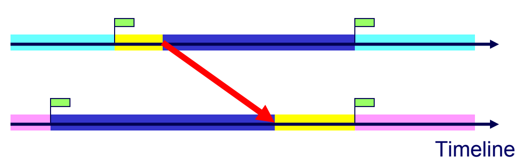
Jesus Labarta, MP,2002

Trace format

Type of record: CPU: thread: Record specific information

Application:task:thread

State	**time_start, time_end, state**
Event	**time, type, value**
Relation	**src, dst, time_src, time_dst, tag, size**



Jesus Labarta, MP,2002

Trace records (.prv)

```
#Paraver (07/11/02 at 14:08):418619736199_ns:0:1:2(8:0,8:0),3
C:1:0:2:1:2
C:1:1:1:-1
C:1:2:2:1:2
2:0:1:1:1:0:40000001:1
1:0:1:1:1:0:3839439:1
…

1:0:1:1:1:3839439:376844493:15
2:0:1:2:1:11779626:40000001:1
1:0:1:2:1:11779626:15502599:1
2:0:1:2:1:15502599:50000003:31:42000003:429662:42000121:2833:42000205:4658:…
1:0:1:2:1:15502599:376805800:15
…
1:0:1:1:1:377926919:378291200:3
3:0:1:2:1:377938400:378219119:0:1:1:1:377926919:378291200:256:1
2:0:1:2:1:377938400:50000001:1:42000003:31798:42000121:396:42000205:486:…
1:0:1:2:1:377938400:378219119:9
2:0:1:2:1:378219119:42000003:27562:42000121:242:42000205:497:42000314:175:…
1:0:1:2:1:378219119:385980426:1
…
```

Application:process:thread

Type and value

Size and tag

State

Jesus Labarta, MP,2002

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
```

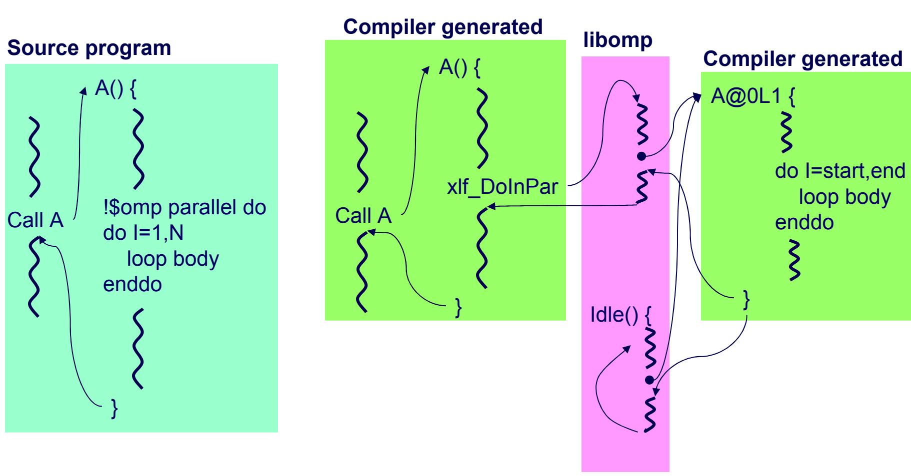
```
EVENT_TYPE
0    60000018     Parallel Function
VALUES
0      End
28      __areamod_MOD_areaave@OL@2
29      __areamod_MOD_areaave@OL@1
30      __atm_lndmod_MOD_atmlnd_drv@OL@2
31      __atm_lndmod_MOD_atmlnd_drv@OL@1
32      camice@OL@1
…
EVENT_TYPE
0   50000001    MPI Point-to-point
VALUES
1   MPI_Send
2   MPI_Recv
3   MPI_Isend
4   MPI_Irecv
6   MPI_Waitall
41   MPI_Sendrecv
0   End
…
EVENT_TYPE
0  42000003 Instructions dispatche
…
0  42000205 Load miss occurred in L1
…
0  42000512 Processor cycles
…
0  42000710 Load instr dispatched
```

Jesus Labarta, MP,2002

Instrumentation

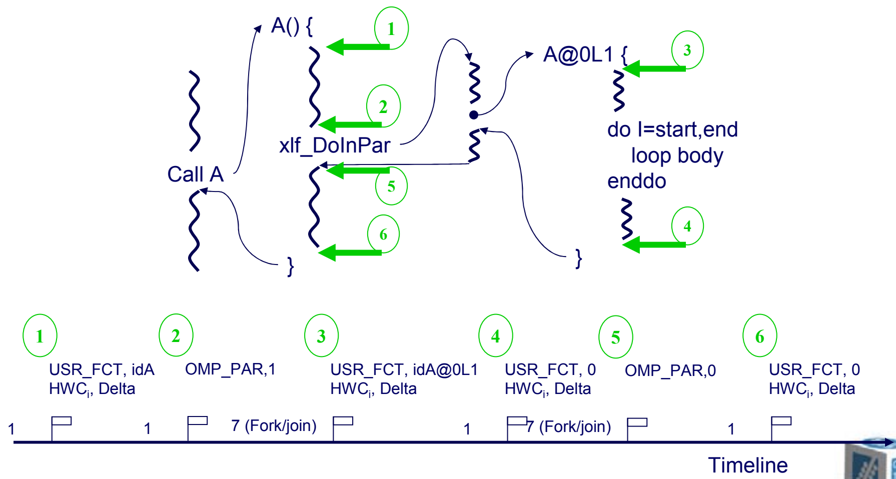
- **Probes**
  - Timing
  - Hardware counters
    - ✓ Portable: PAPI (http://icl.utk.edu/projects/papi)
    - ✓ Vendor specific: PMAPI, …
- **Insertion of probes**
  - Dynamic
    - ✓ DItools (http://personals.ac.upc.es/alberts/fpc.html)
    - ✓ DPCL (http://oss.software.ibm.com/developerworks/opensource/dpcl)
      - – IBM, Linux?
      - – Dyninst based (http://www.dyninst.org/)
  - Static:
    - ✓ PMPI

OpenMP compilation and Run Time



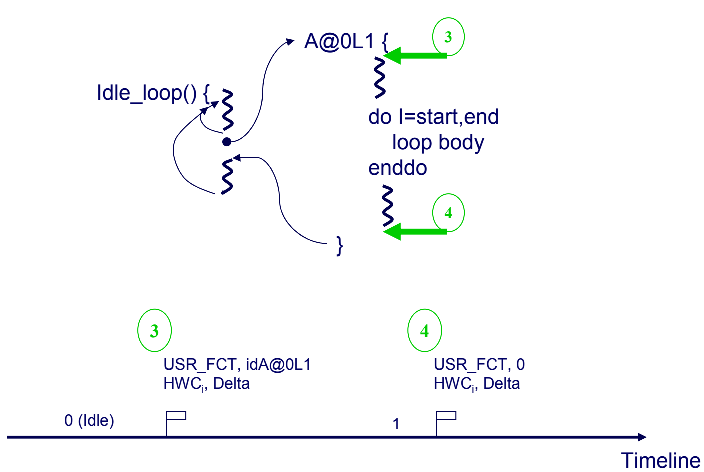
OpenMP instrumentation points

- **Main thread**



OpenMP instrumentation points

- **Slave threads**



Linux Instrumentation : MPI

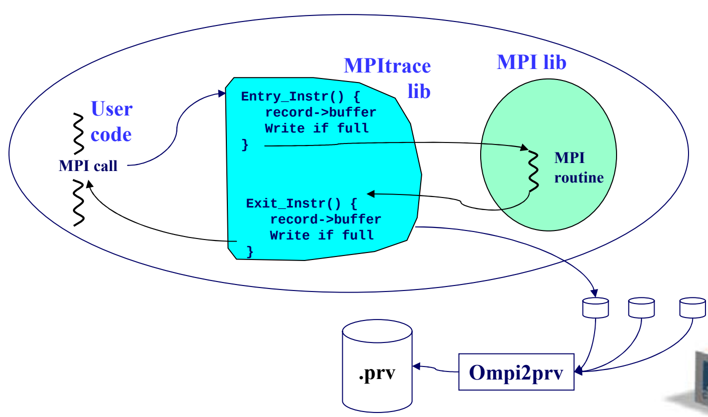
- Using PMPI interface: Events at entry/exit of MPI calls
- PAPI: include hardware counter events
- MPItrace API: include user callable routines to
  - emit events
    - ✓ mpitrace_event(int type, int value)
    - ✓ mpitrace_eventandcounters(int type, int value)
    - ✓ mpitrace_counters()
  - stop/resume tracing
    - ✓ mpitrace_shutdown(), mpitrace_resume()

Linux tracing

- Use

  setenv MPTRACE_COUNTERS <list_of_hwc_events>

  mpitrace mpirun -np <nb procs> -machinefile <hosts> myprogram



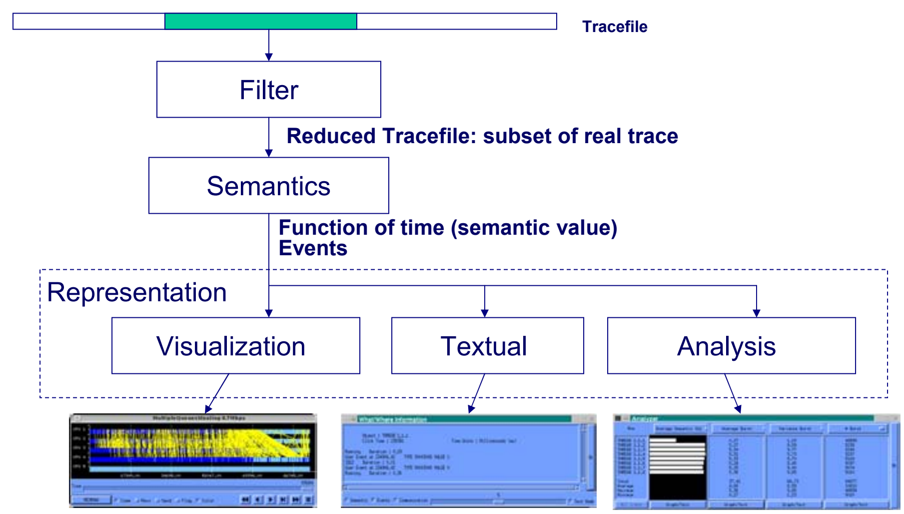
Overhead

- **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

Jesus Labarta, MP,2002

Structure



Demand Driven evaluation

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

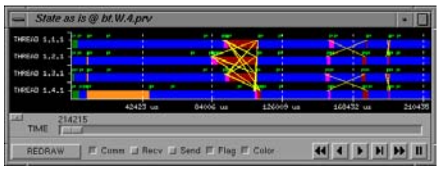
- **What : restrict records that pass to the semantic module**
  - Events
    - ✓ by type
    - ✓ by value
  - Communications
    - ✓ by tag
    - ✓ by size
    - ✓ by source / destination
    - ✓ logical / physical
- **What for**
  - Reduce amount of information to display
  - Feed properly the semantic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

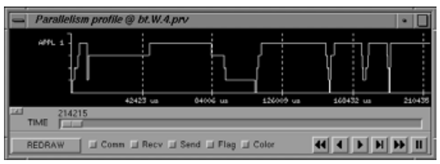
Filter module

- **Display window to which applies**
- **Communications that pass through the filter**
- **Show list of tasks**
- **events that pass through the filter**
- **Show list of event types and values**

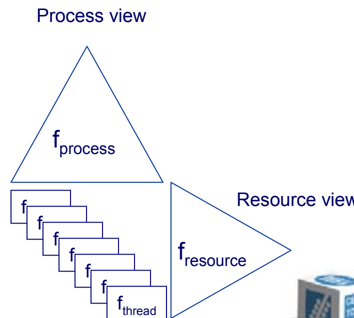
- **Dismiss filter window**

Semantic module

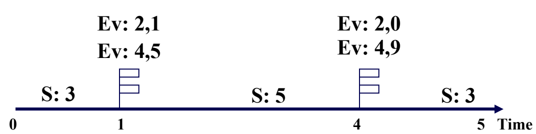




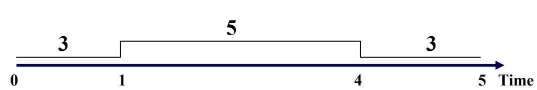
- **Angle:**
  - Process model
    - ✓ Thread, task, application, workload
  - Resource model
    - ✓ CPU, node, system



Semantic module: Thread function

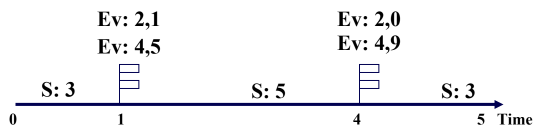


- **Thread function: State as is**



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

Semantic module: Thread function



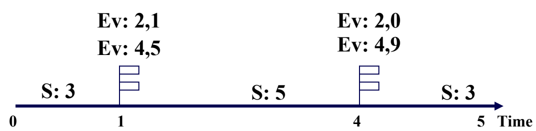
- **Filter: type == 2**

  Thread function: Last event value

- **Useful for**
  - In parallel region
  - Mutual exclusion
  - Variable values: iteration,….

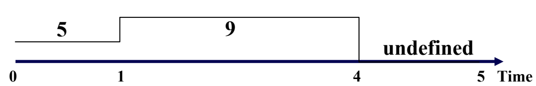
Jesus Labarta, MP,2002

Semantic module: Thread function



- **Filter: type == 4**

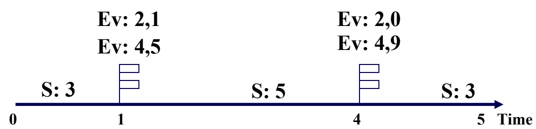
  Thread function: Next event value



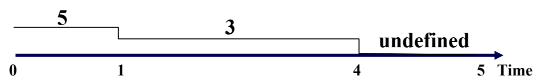
- **Useful for**
  - Hwc events (TLB, L1 misses,...) within interval

Jesus Labarta, MP,2002

Semantic module: Thread function



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

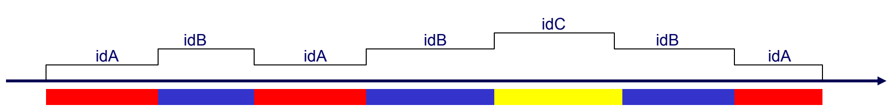


- **Useful for**
  - Hwc events (TLB, L1 misses,...) per time unit within interval

Jesus Labarta, MP,2002

Semantic module: Thread function

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

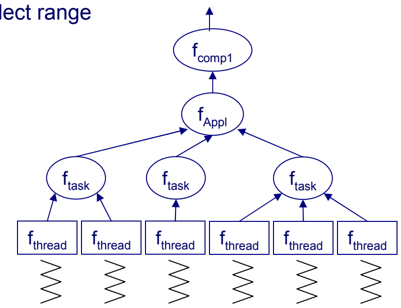


- **Useful for**
  - Routine

Jesus Labarta, MP,2002

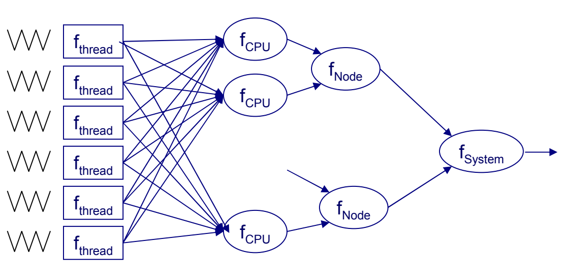
Semantic module: process model view

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



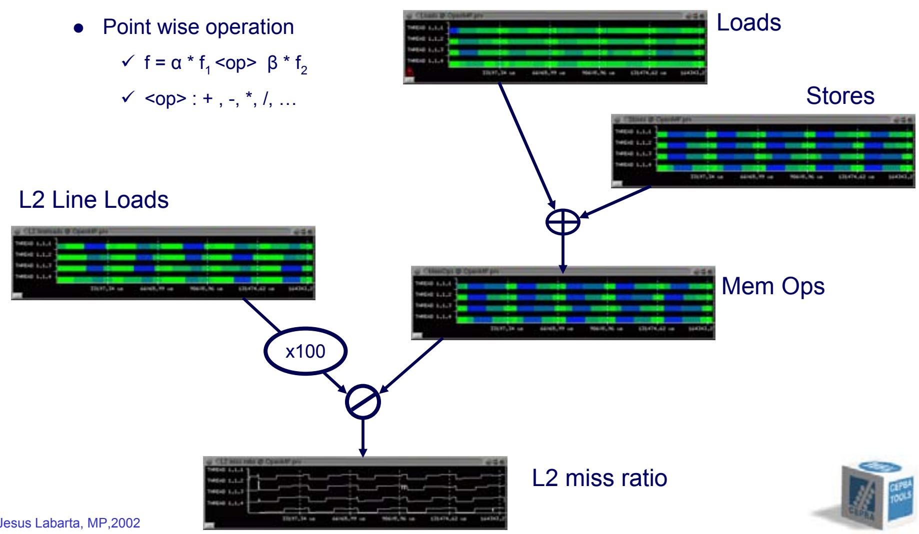
Semantic module: resource view

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



Semantic module

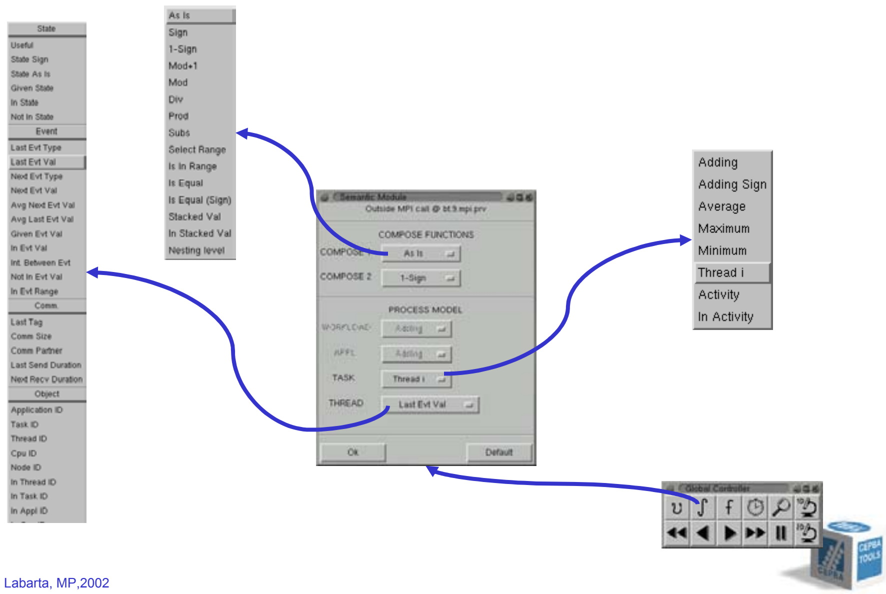
- **Derived windows**
  - Point wise operation
    - ✓ $f = \alpha * f_1$ <op> $\beta * f_2$
    - ✓ <op> : + , -, *, /, …



Semantic module

- **Derived windows**
  - Point wise operation
    - ✓ $f = \alpha * f_1$ <op> $\beta * f_2$
    - ✓ <op> : + , -, *, /, …

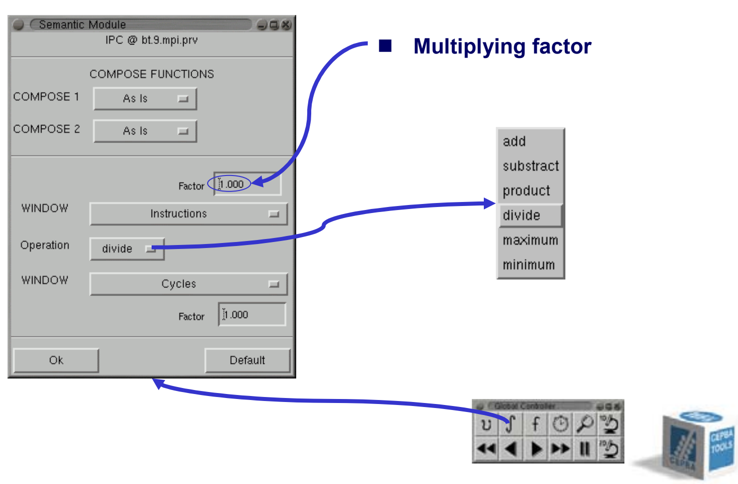
Semantic module: process model view



Jesus Labarta, MP,2002

Semantic module: derived windows

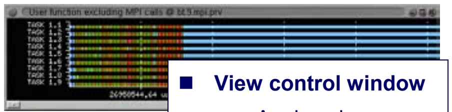
- How to build expression



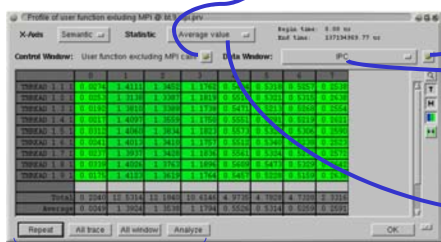
- Multiplying factor

Jesus Labarta, MP,2002

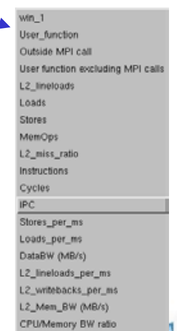
2D analysis

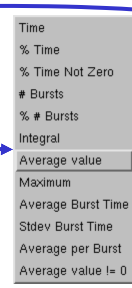


- View control window
  - Analyzed area
- View data window



- Select data window





- Select statistic

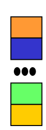
- Perform an analysis
  - On same area
    - On whole trace (same CW)
      - On whole CW
        - Selecting new CW

Jesus Labarta, MP,2002

2D analysis

- Control window bin definition
  - Min/max/delta
  - Fit to generate <= 20 columns

- Statistic color encoding
  - Min/max



max

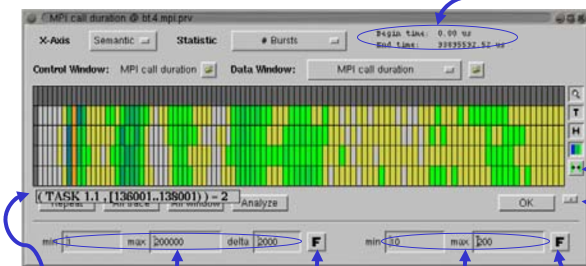
min

  - Fit Min/max value

Jesus Labarta, MP,2002

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

  max

  •••

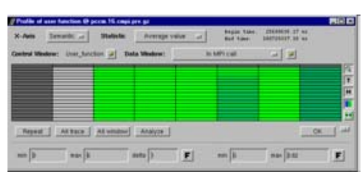
  min
- Fit color encoding

  Min and max to dynamic range of statistic

Jesus Labarta, MP,2002

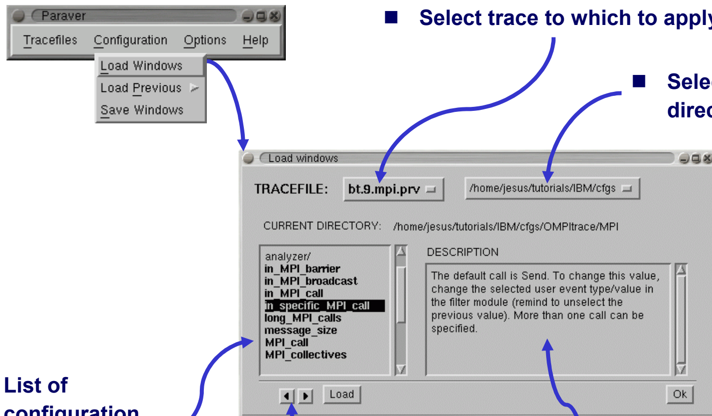
2D analysis

- **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



Jesus Labarta, MP,2002

Configuration files

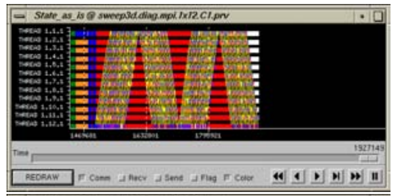


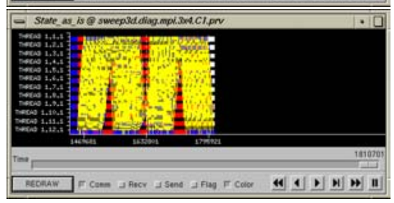
- Select trace to which to apply
- Select directory
- List of configuration files in current directory
- Navigate through directory tree
- Description of view of select file

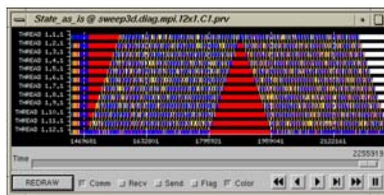
Some examples

Understanding applications/system

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

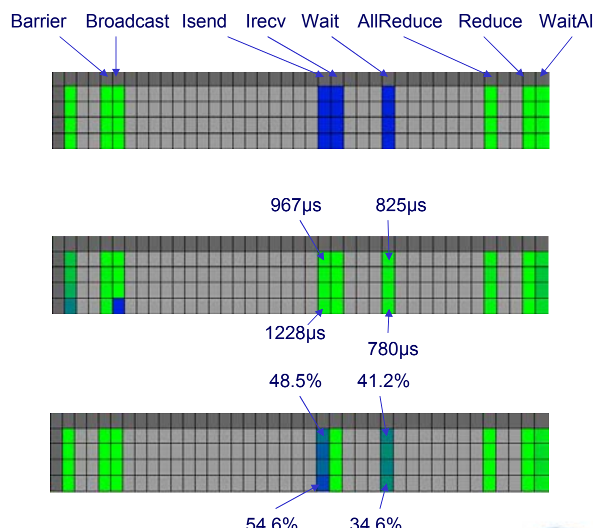






Understanding applications/system

- **MPI calls analysis**
  - # MPI calls
  - Average MPI call duration
  - Percentage of time in MPI call (vs. Total time in MPI calls)

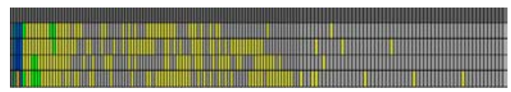


Understanding applications/system

- **Isend duration histogram**

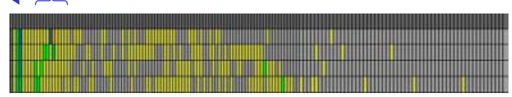
Blue 30%
Green 3%

  - % Number of calls



100-600 µs
1200-3000µs

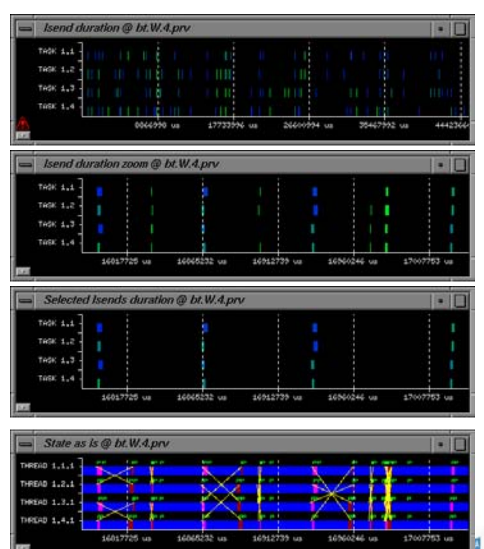
  - Total time



0ms
Δ =200µs
30ms

Understanding applications/system

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



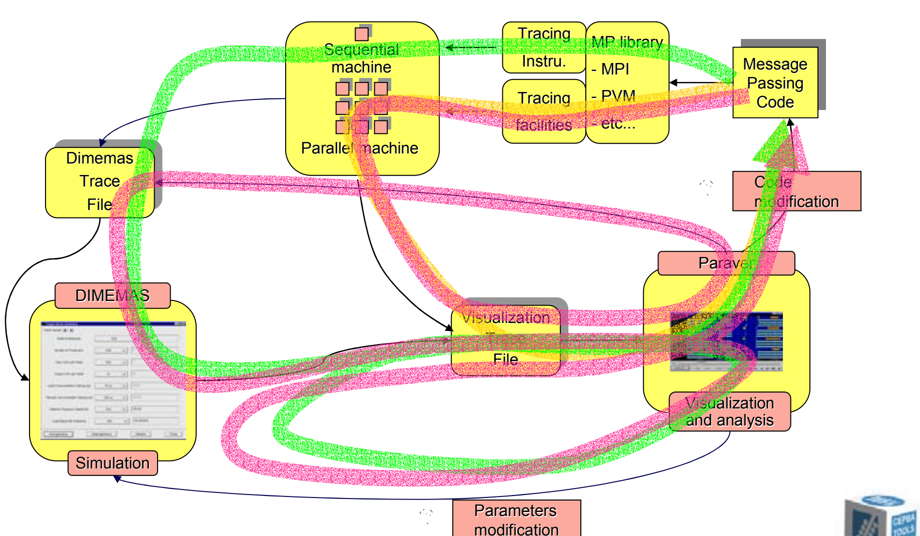
Dimemas overview

Jesús Labarta, Judit Gimenez
Jordi Caubet, Francesc Escale

CEPBA-UPC

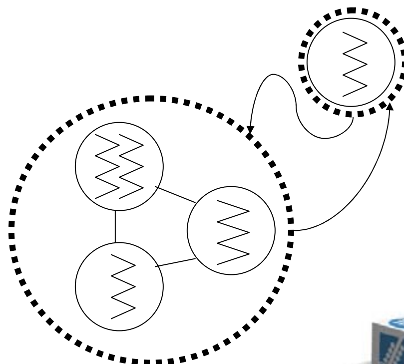
Technology Transfer   Research   Training   Mobility of Researchers
User Support   Education   HPC Facilities   Parallel Expertise

Dimemas



Tracefile

- **Characterises application**
  - Sequence of resource demands for each task
  - Sequence of events: communication
- **Application model**



Jesus Labarta, MP,2002

Tracefile

- **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_recvd";
};;
```

Jesus Labarta, MP,2002

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, 3640, 10003, 0, 0 };;
"block end" { 35, 0, 75 };;
…
```

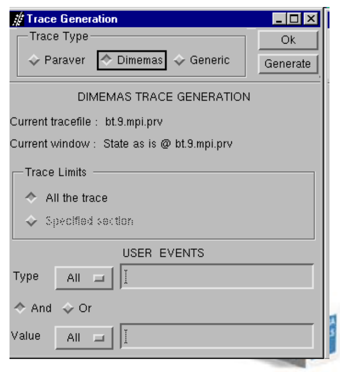

Tracefile

- **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

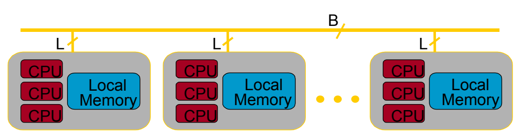
- **Dimemas instrumentation**
  - MPIDtrace
    - ✓ Run the same way as OMPItrace
- **Tracefile generated by OMPItrace**
  - Paraver: ->Trace Generation
    - ✓ The Paraver trace should have been obtained with dedicated resources
    - ✓ Does not support selection of specific part of the trace
    - ✓ Possible to get rid of non relevant events



Jesus Labarta, MP,2002

Parallel machine model

- **Network of SMPs**
- **Multiprogrammed workload**



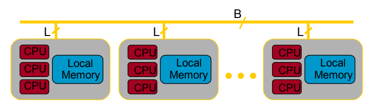
- **Key factors influencing performance**
  - “Abstract” architecture
  - Basic MPI protocols
  - No attempt to model details of a specific implementation
- **Objectives**
  - Simple/general
  - Fast simulation

Jesus Labarta, MP,2002

p2p communication model

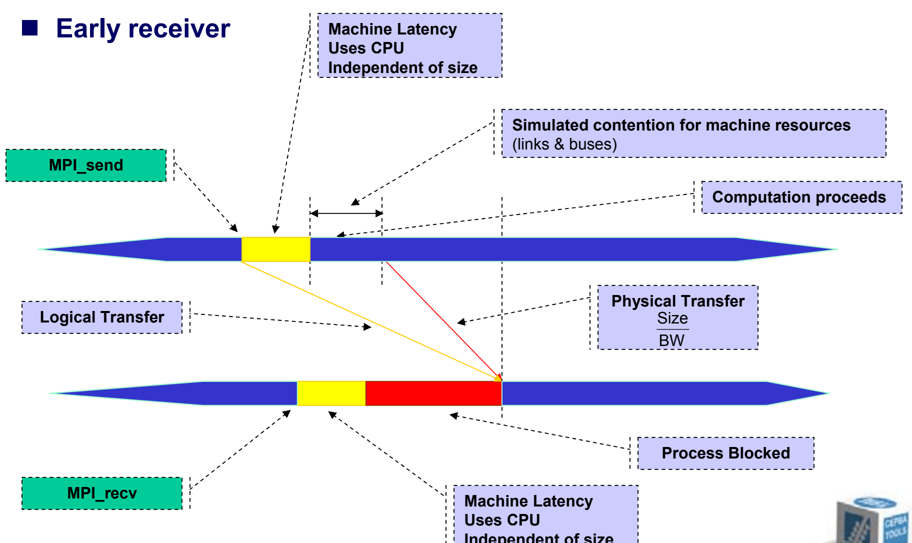
- **Latency**
  - Depends on node. Independent of message size
  - Paid by sends and receives at beginning of operation
  - Uses CPU
- **Network Bandwidth**
  - Independent of message size
  - Uses links and buses
    - ✓ Links:
      - – determine concurrent accesses to the network
      - – Half / Full Duplex
    - ✓ Busses:
      - – maximum number of simultaneous transfers
    - ✓ Output link, input link and bus must be allocated to start transmission

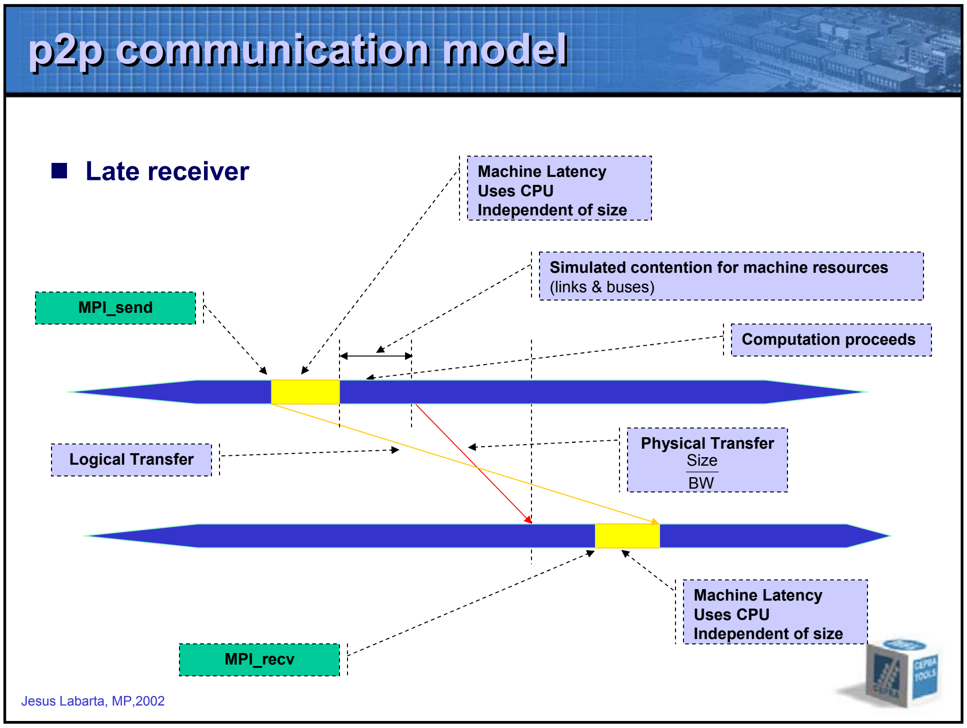
$$T = Latency + \frac{msg.size}{Bandwidth}$$

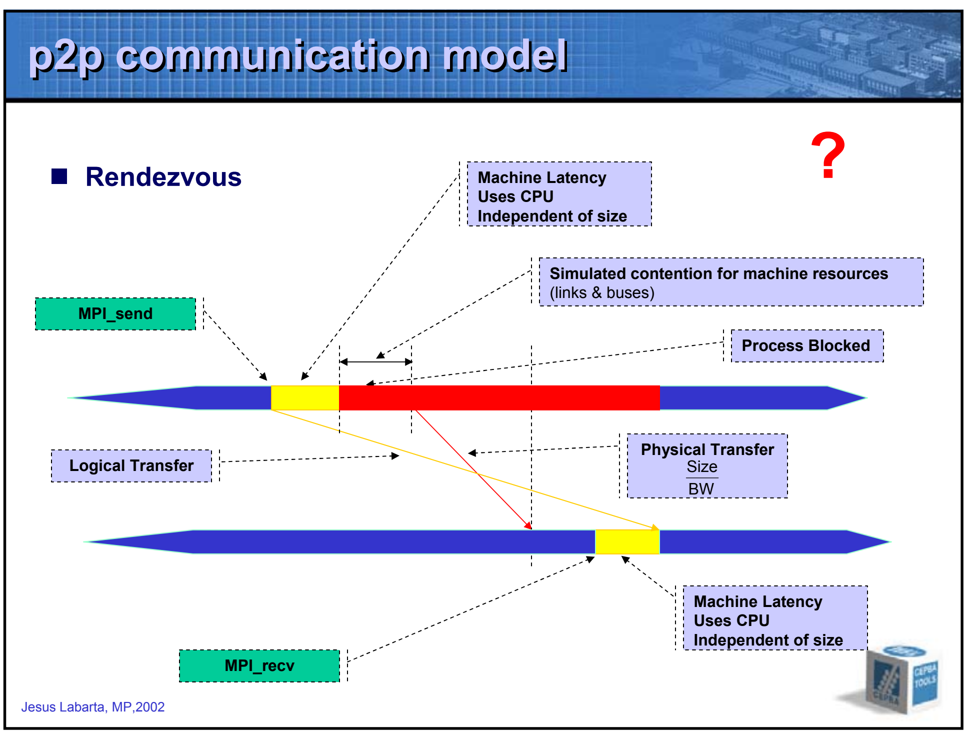


p2p communication model

- **Early receiver**

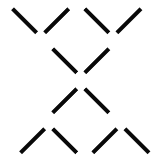


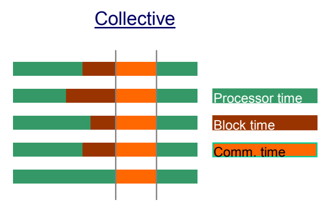

p2p communication model
Late receiver
Machine Latency
Uses CPU
Independent of size
Simulated contention for machine resources
(links & buses)
MPI_send
Computation proceeds
Logical Transfer
Physical Transfer
Size
BW
Machine Latency
Uses CPU
Independent of size
MPI_recv
Jesus Labarta, MP,2002


p2p communication model
Rendezvous
?
Machine Latency
Uses CPU
Independent of size
Simulated contention for machine resources
(links & buses)
MPI_send
Process Blocked
Logical Transfer
Physical Transfer
Size
BW
Machine Latency
Uses CPU
Independent of size
MPI_recv
Jesus Labarta, MP,2002

Collective communication model

- **Phases**
  - Barrier
  - Fan-in
  - Fan-out
- **Cost of communication phase**
  - ✓ Generic
  - ✓ Per call

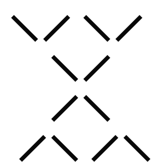


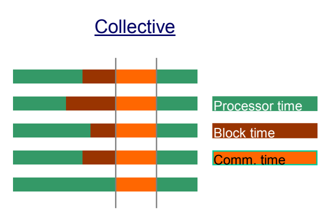


Jesus Labarta, MP,2002

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





Jesus Labarta, MP,2002

Collective Communication Model

- **Generic model**
  - Communication time

$$\text{Time} = \left(\text{Latency} + \frac{\text{Size}}{\text{Bandwidth}}\right) * \text{MODEL\_FACTOR}$$

  - Model factor
    - ✓ Lin / log / const

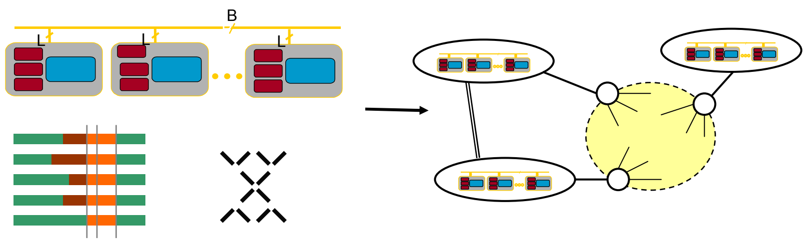
Model	Factor
Null	0
Constant	1
Linear	P
Logarithmic	$\text{Nsteps} = \sum_{i=1}^{\lceil \log_2 P \rceil} \text{steps}_i, \text{steps}_i = \left\lceil \frac{C}{B} \right\rceil$

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

	IN		OUT	
Operation	**Model**	**Size**	**Model**	**Size**
Barrier	LIN	MAX	LIN	MAX
Bcast	LOG	MAX	NULL	
Gather	LOG	MEAN	NULL	
Gatherv	LOG	MEAN	NULL	
Scatter	NULL		LOG	MEAN
Scatterv	NULL		LOG	MEAN
Allgather	LOG	MEAN	LOG	MEAN
Allgatherv	LOG	MEAN	LOG	MEAN
Alltoall	LOG	MEAN	LOG	MAX
Alltoallv	LOG	MEAN	LOG	MAX
Reduce	LOG	2MAX	NULL	
Allreduce	LOG	2MAX	LOG	MAX
Reduce_Scatter	LOG	2MAX	LOG	MIN
Scan	LOG	MAX	LOG	MAX

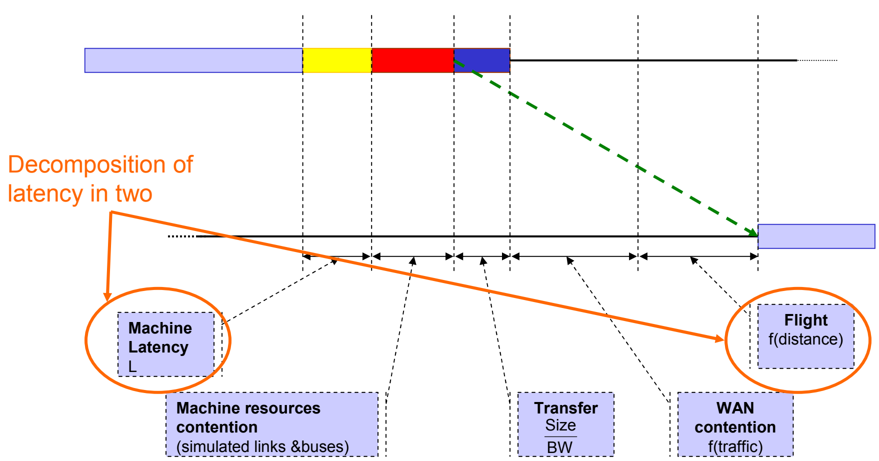
Dimemas GRID: model extension



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

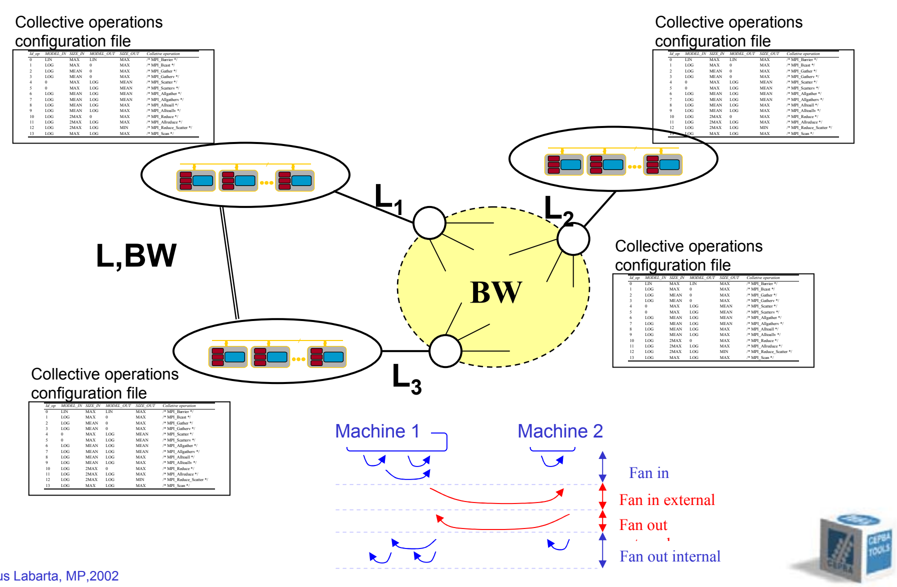
Jesus Labarta, MP,2002

Dimemas GRID: communication model



Jesus Labarta, MP,2002

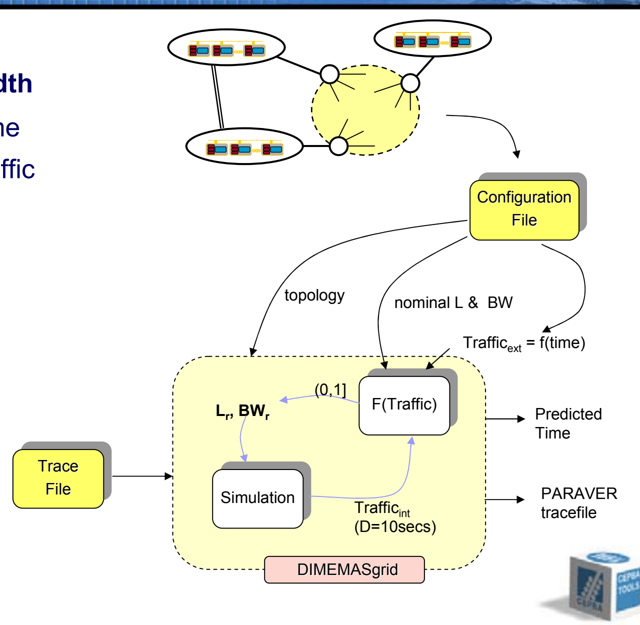
Collective operation model



Jesus Labarta, MP,2002

Dimemas GRID: other features

- **Variable bandwidth**
  - Function of time
  - Function of traffic



Jesus Labarta, MP,2002

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 tranfer rate between nodes in Mbytes/s"
          // "0 means instantaneous communication"
          double  "network_bandwidth";
          // "number_of_buses_on_network" "Maximun number of messages on network"
          // "0 means no limit"
          // "1 means bus contention"
          int       "number_of_buses_on_network";
          // "1 Constant, 2 Lineal, 3 Logarithmic"
          int       "communication_group_model";
};;
…
```


Architecture description

- **Configuration file**

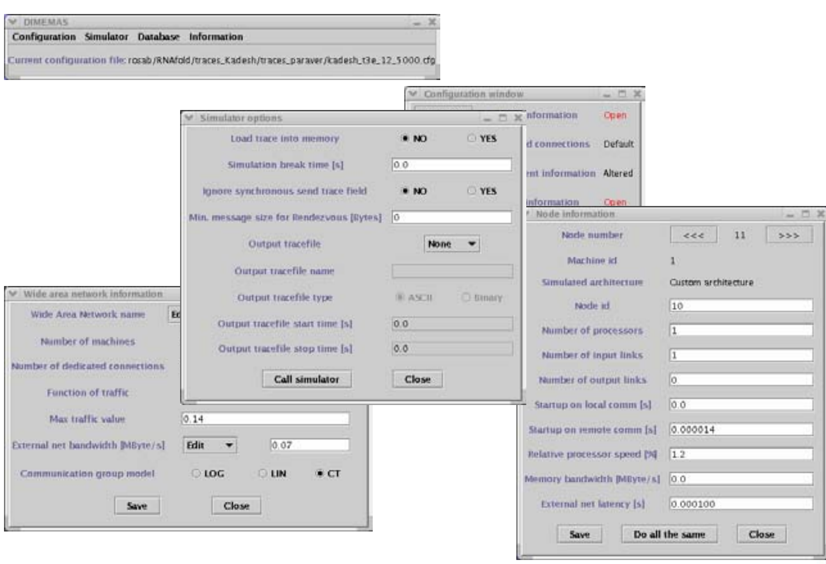
```
…
#2:
"node information" {
          int "machine_id";
          // "node_id" "Node number"
          int     "node_id";
          // "simulated_architecture" "Architecture node name"
          char    "simulated_architecture"[];
          // "number_of_processors" "Number of processors within node"
          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 tranfer rate into node in Mbytes/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};;

"environment information" {"", 3, "", 1, 0.0, 0, 1};;
"environment information" {"", 2, "", 1, 0.0, 0, 1};;
"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, 0.0, 1.0, 0.0, 0.0};;
"node information" {1, 1, "", 1, 1, 1, 0.0, 0.0, 1.0, 0.0, 0.0};;
"node information" {2, 2, "", 1, 1, 1, 0.0, 0.0, 1.0, 0.0, 0.0};;
"node information" {3, 3, "", 1, 1, 1, 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};;
…
```

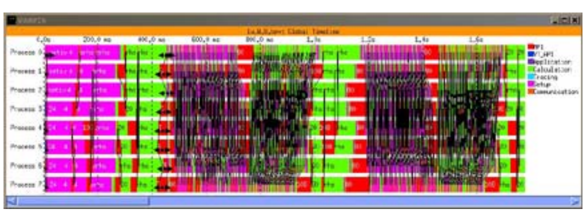

Dimemas GUI



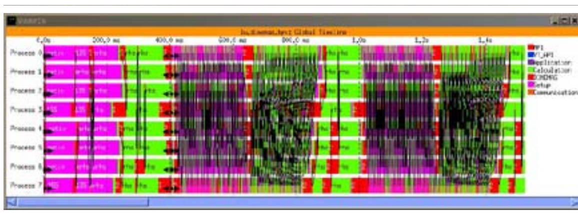
The simulator: Dimemas

Jesus Labarta, MP,2002

Qualitative validation



Dedicated machine



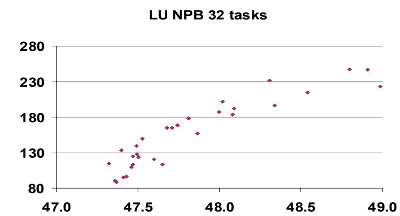
Shared environment

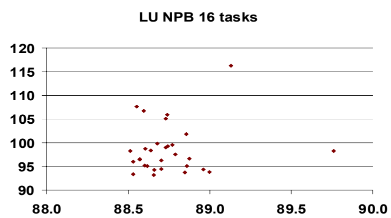
+ Dimemas

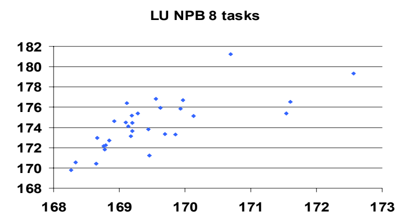
Jesus Labarta, MP,2002

Stability validation

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





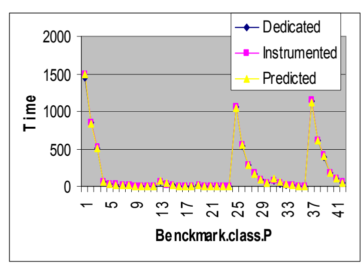


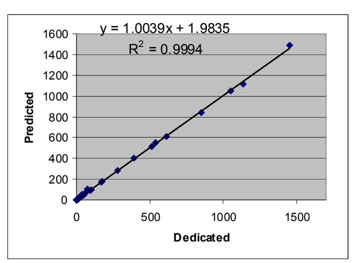
Jesus Labarta, MP,2002

NAS Benchmarks

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

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





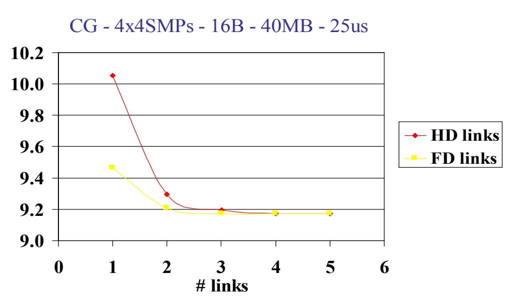
Jesus Labarta, MP,2002

Application Analysis

- **Load balanced and dependence problems?**
  - BW = ∞, L = 0
- **Group messages?**
  - L = … , BW = ∞
- **Bandwidth problem?**
  - BW = … , L = 0
- **Concurrent communication problems?**
  - BW = target , L = target , buses = 1, 2, ...

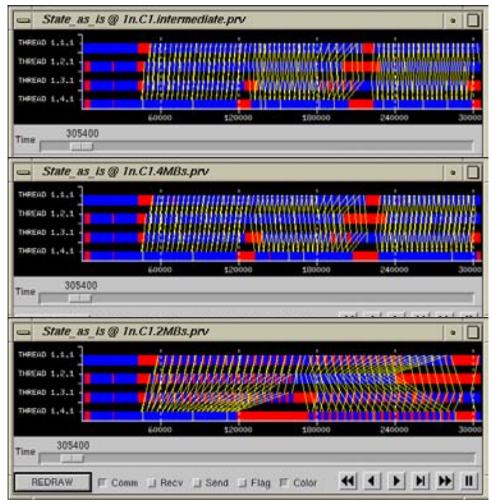
Bandwidth Trade-offs

- **Injection mechanism**



Bandwidth Trade-offs

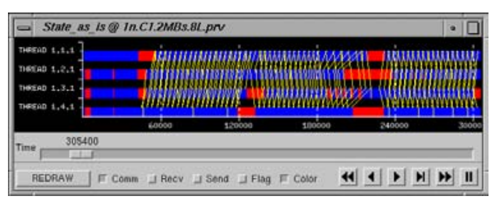
- L=50 µs , BW= 8 MB/s
- L=50 µs , BW= 4 MB/s
- L=50 µs , BW= 2 MB/s



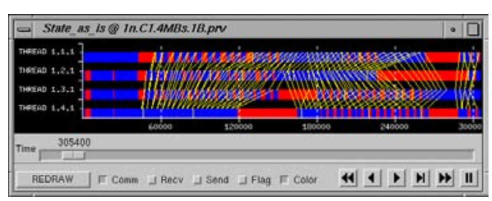
===> 2 MB/s is a problem …

Bandwidth Trade-offs

- L=50 µs , BW= 2 MB/s but 8 links to network per node

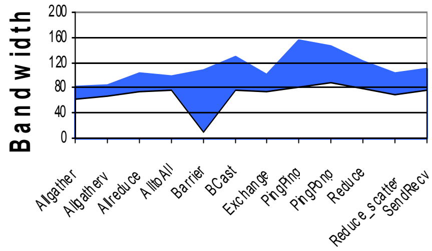


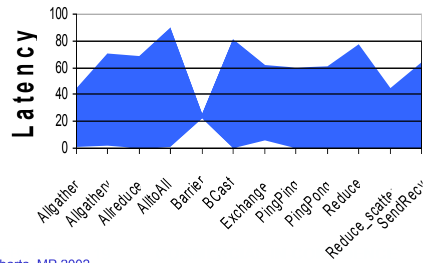
- L=50 µs , BW= 4 MB/s but 1 bus



… trade off raw bandwidth - bisection/connection BW

System characterization



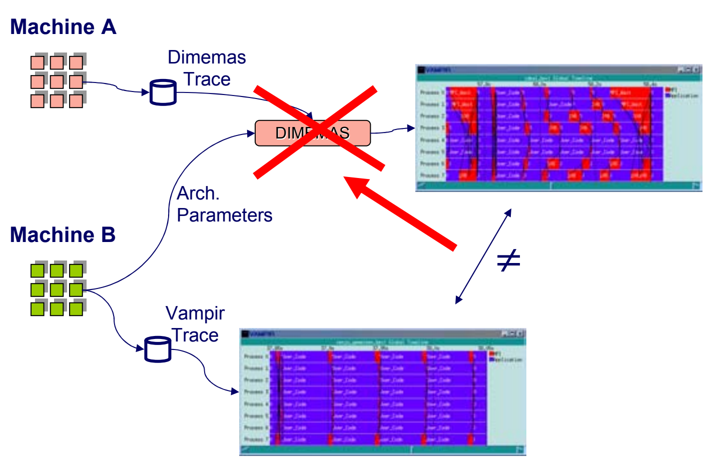


< 10% error regions

1 half duplex link !!!

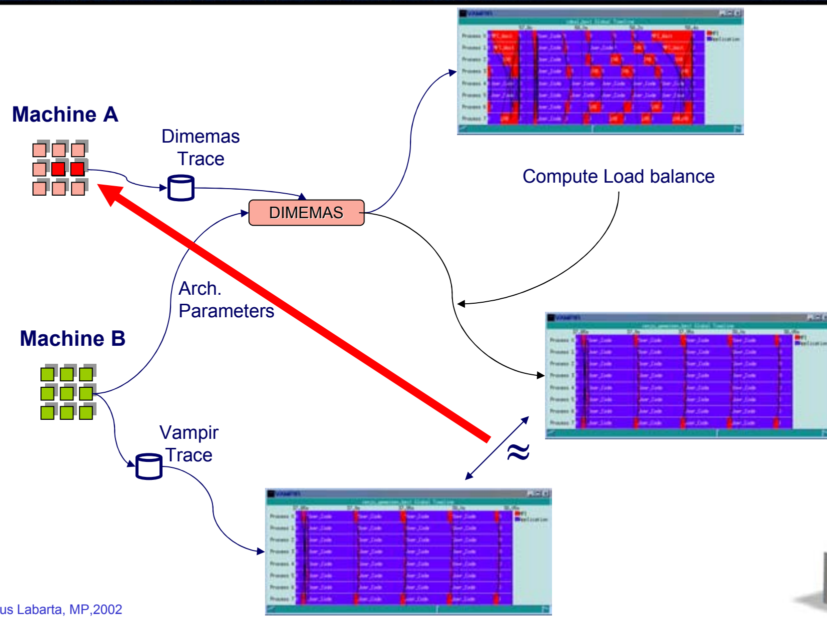
Jesus Labarta, MP,2002

Understanding architectures



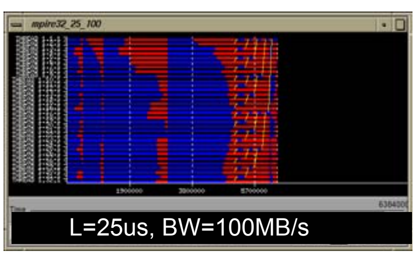
Jesus Labarta, MP,2002

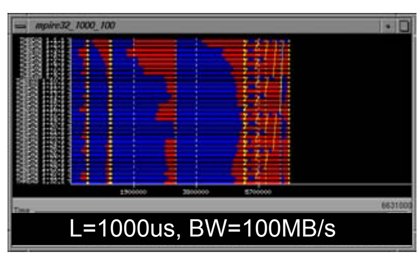
Understanding architectures

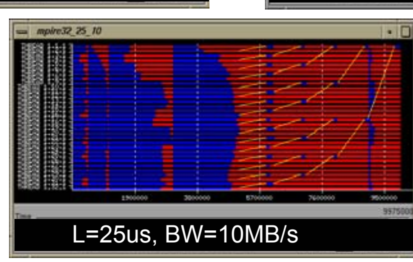


Understanding applications (MPIRE)

- **32 procs (no network contention)**



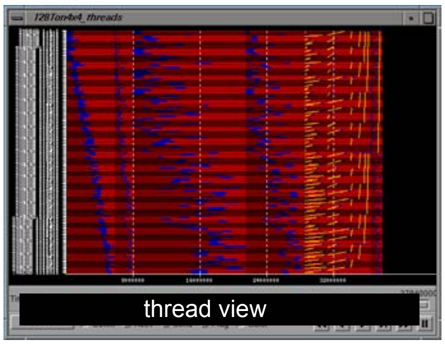


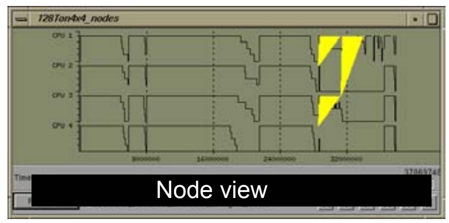


All windows same scale

Understanding applications (MPIRE)

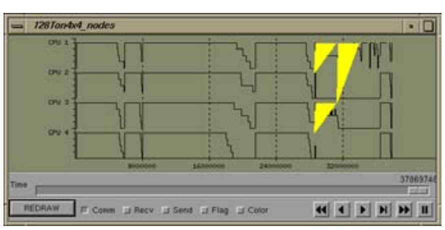
- **Cluster of SMPs**
  - 4nodesx4, 1 link
  - 128 p, L=25, BW=100, no network contention

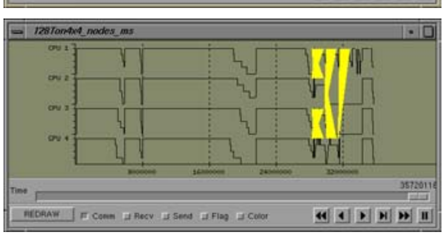


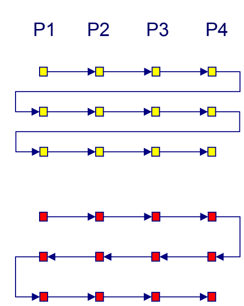


Understanding applications (MPIRE)

- **Mapping influence**
  - 128 p, L=25, BW=100, 4nodesx4, 1 link, no network contention

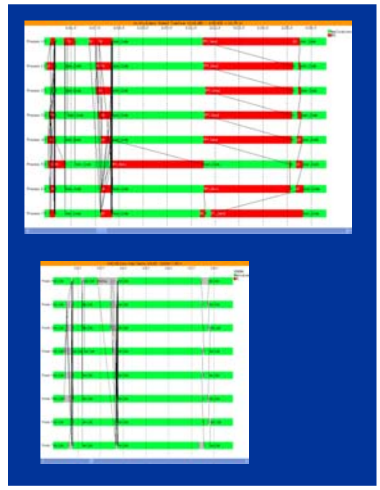






Understanding architectures

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



* Courtesy FECIT

Jesus Labarta, MP,2002

Sensitivity analysis

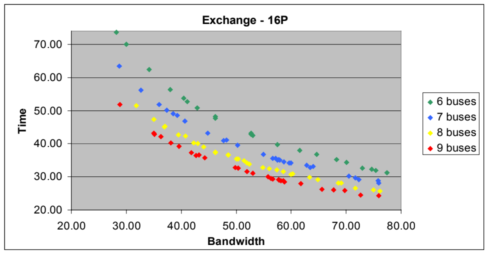
- **How sensitive is my program to bandwidth, latency, injection mechanism, routine optimization?**
  - How much do they influence the execution time?
    - ✓ ST-ORM: “GRID” tool for specifying studies (montecarlo, optimization,parametric studies…), generating and submitting the jobs, collecting and 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?**

Jesus Labarta, MP,2002

Montecarlo studies

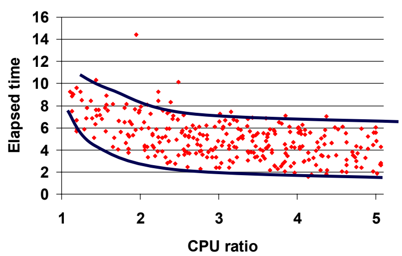
- **Contention**

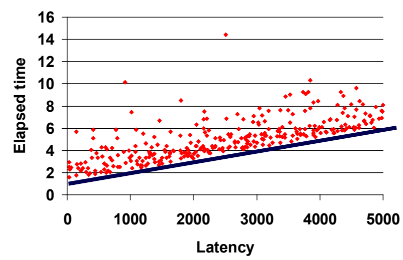


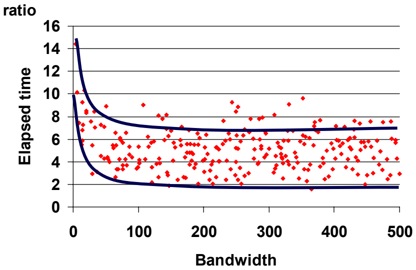
Jesus Labarta, MP,2002

Montecarlo studies

- **MPIRE: 32 CPUs (no network contention)**



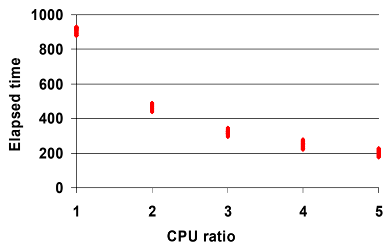


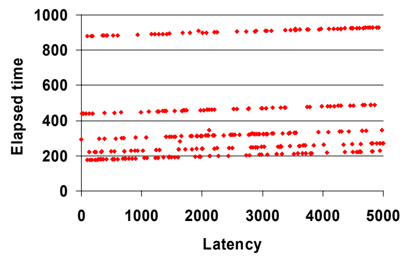


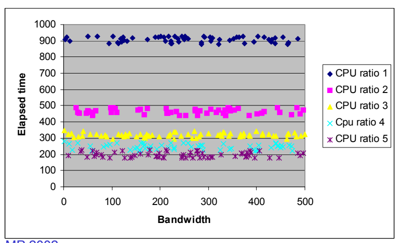
Jesus Labarta, MP,2002

Montecarlo studies

- **SCF: 32 CPUs (no network contention)**

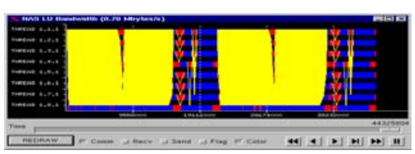


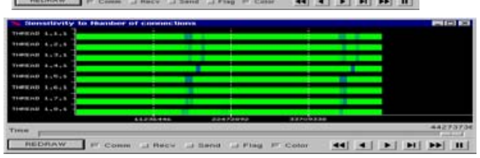




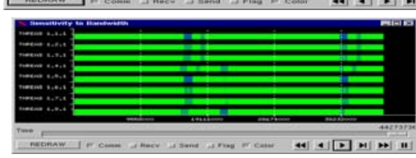
Essentially Sensitive to raw processor performance

Program section sensitivity

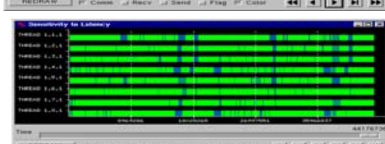




to contention



to bandwidth



to latency

Metasim + Dimemas

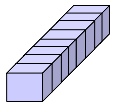
- **Convolution**
  - Observed performance is a convolution of
    - ✓ algorithm characteristics
      - Instructions
      - Communication demands
    - ✓ machine characteristics
      - Processor
      - Communication
- **Tracing is an attempt to deconvolve the algorithm and machine characteristics from an actual run to later convolve (Metasim/Dimemas) with the hypothetical target machine characteristics**

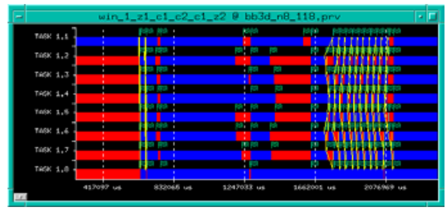
Metasim + Dimemas

- **Cooperation with Allan Snavelly (SDSC)**
- **Estimate CPU ratios between tracing and target machine**
  - Based on simple instruction level simulation
  - Started by simple hypothesis
    - ✓ IPC $\propto$ bandwidth

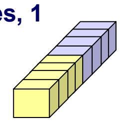
Dimemas GRID: example

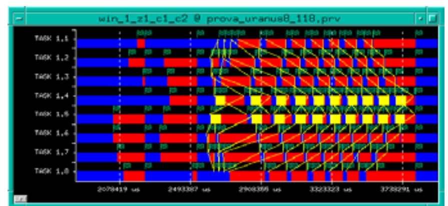
- URANUS
- 1 machine, 8 nodes, 1 proc/node





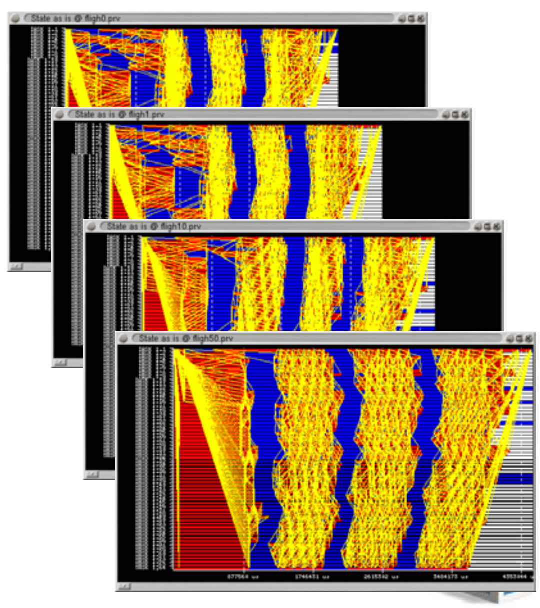
- latency=30 msecs, BW=0.8 MB/S
- 2 machines, 4 nodes, 1 processor/node





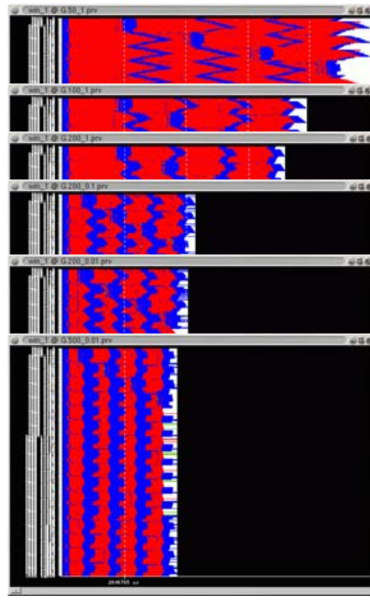
Example: Uranus

- 64 processes
- 4 16-way SMPs
- Fligth times
  - 0,1,10 and 50 ms.



Example: linpack

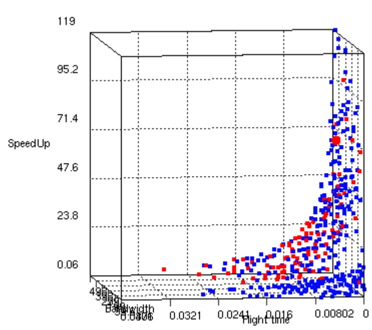
- **256 processes**
- **16 16-way SMPs**
- **BW (MB/s) / Flight time (ms)**
  - 50 / 1
  - 100 / 1
  - 200 / 1
  - 200 / 0.1
  - 200 / 0.01
  - 500 / 0.01

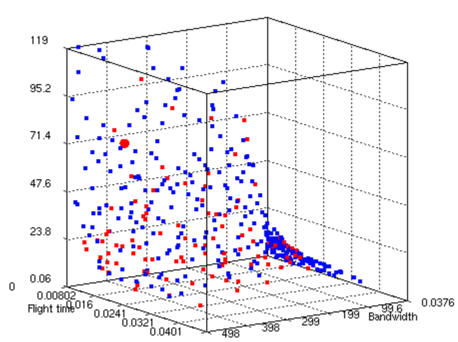


Jesus Labarta, MP,2002

Example: Explore response surface

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





Jesus Labarta, MP,2002

RNAfold results*

Machines involved:

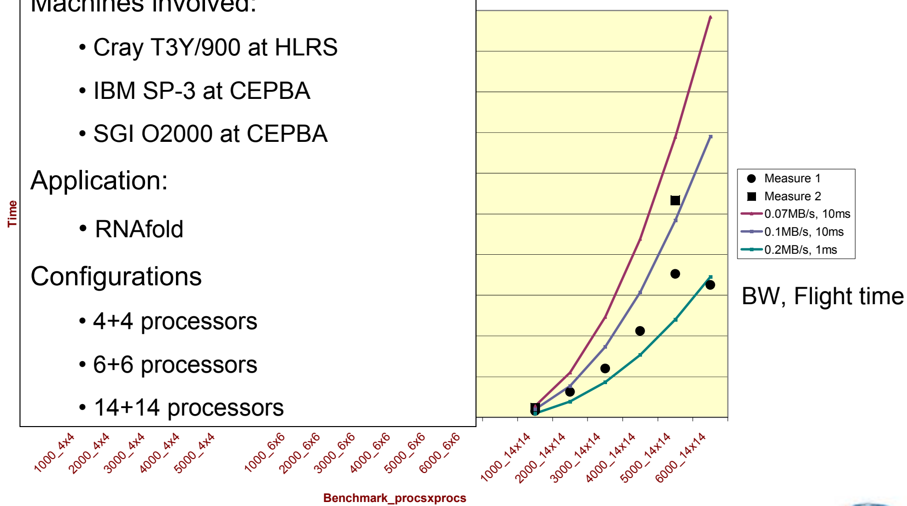
- Cray T3Y/900 at HLRS
- IBM SP-3 at CEPBA
- SGI O2000 at CEPBA

Application:

- RNAfold

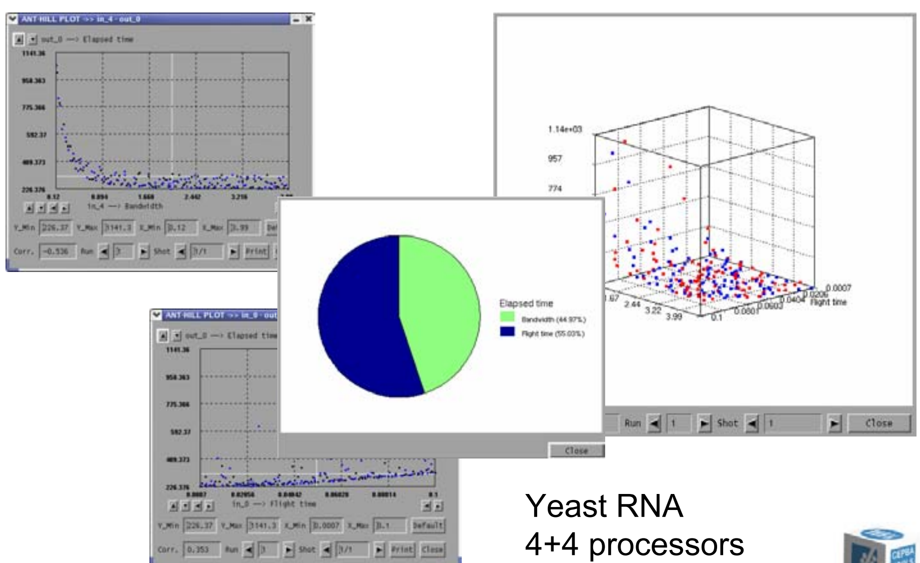
Configurations

- 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

More information

http://www.cepba.upc.es/dimemas

cepbatools@cepba.upc.es