Barcelona Supercomputing Center
Centro Nacional de Supercomputación

CEPBA-Tools

BSC Performance Tools

Performance Analysis Objective

How is my application performing?

Can I describe it in a simple way? Quantitatively?

Is there anything I can do to improve its performance?

What?

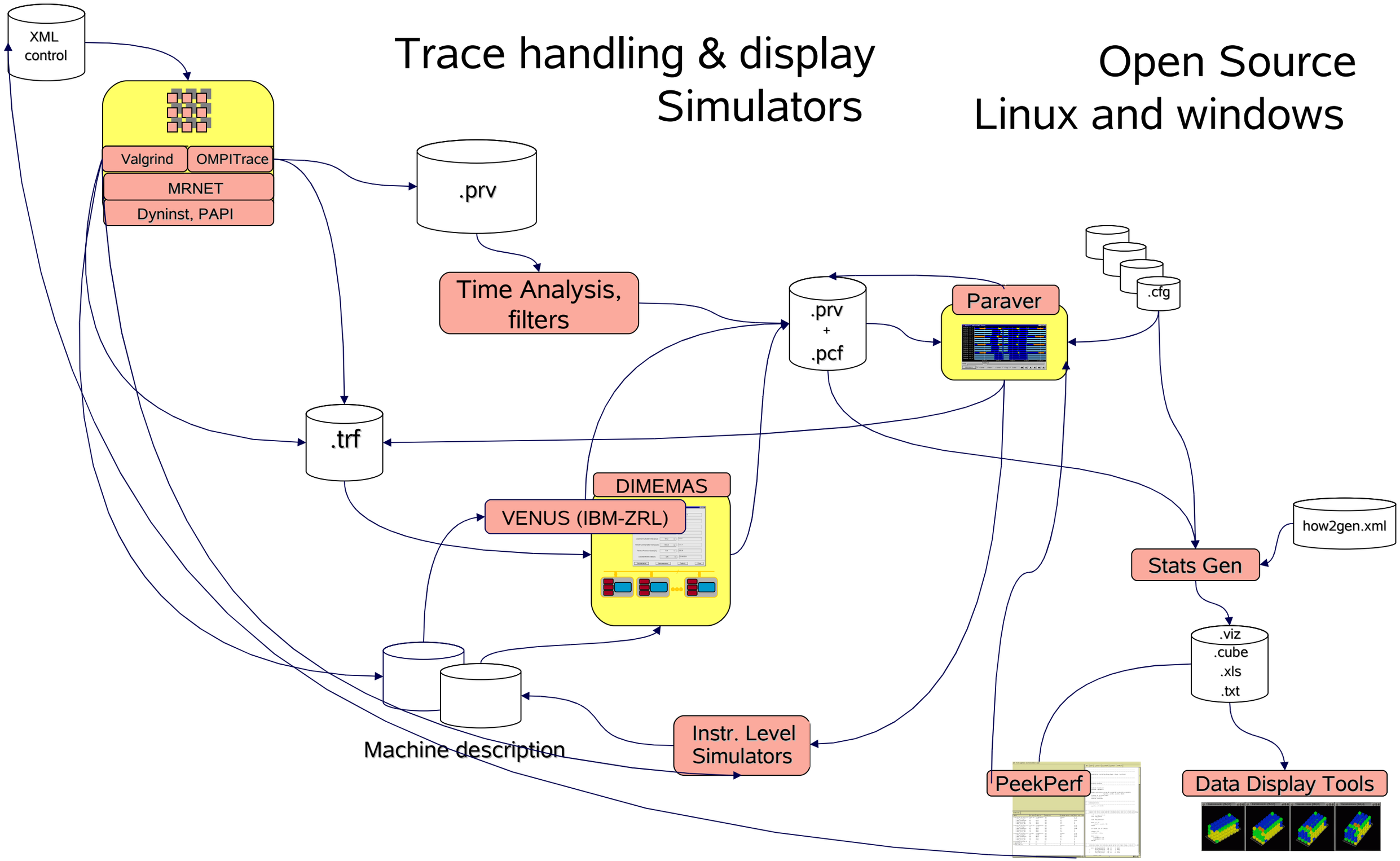
- **Specific**
- **Expected performance improvement**
- **Preferably with minimum effort/cost**

Who can I blame?

CEPBA – tools framework

Trace handling & display
Simulators

Open Source
Linux and windows



Extrae - Instrumentation

Parallel Program Instrumentation: Features

- Probe injection mechanisms
  - Library Preload: Linux clusters (Dynamically linked MPI library)
  - Static linking: BG
  - Compiler-based instrumentation (PDT) ALTIX, PowerPC
  - dynamic instrumentation (Dyninst): ALTIX, PowerPC (beta version)
- Captured events:
  - MPI calls (including I/O)
  - Hardware counters: PAPI: standard + native
    - Several sets: Rotate groups periodically / Different groups per processes
  - Network: GM at end of trace, MX at flushes
  - OS counters: At the end of the trace and when flushing
  - Link to source:
    - Dyninst based systems: user function events on entry and exit (for selected functions).
    - Library preload: MPI caller (several levels)
  - User events (API)
- Towards a unified tracing package and specification
  - xml control file

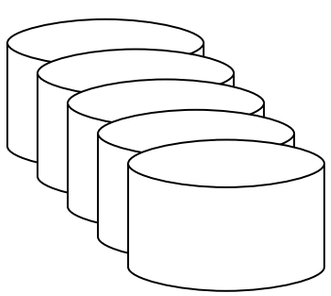
A typical MareNostrum process

mn_trace

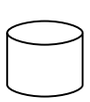
```
#!/bin/bash
…
# @ total_tasks = 2
# @ cpus_per_task = 1
# @ tasks_per_node = 2
…
srun ./trace.sh ./mpi_ping
```

trace.sh

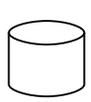
```
export#!/bin/sh

export MPITRACE_HOME=/gpfs/apps/CEPBATOOLS/mpitrace/64
export MPTRACE_CONFIG_FILE=mpitrace.xml

export LD_PRELOAD=${MPITRACE_HOME}/lib/libmpitrace.so

## Run the desired program
$*
```



TRACExxxxxx.mpit



TRACE.mpits



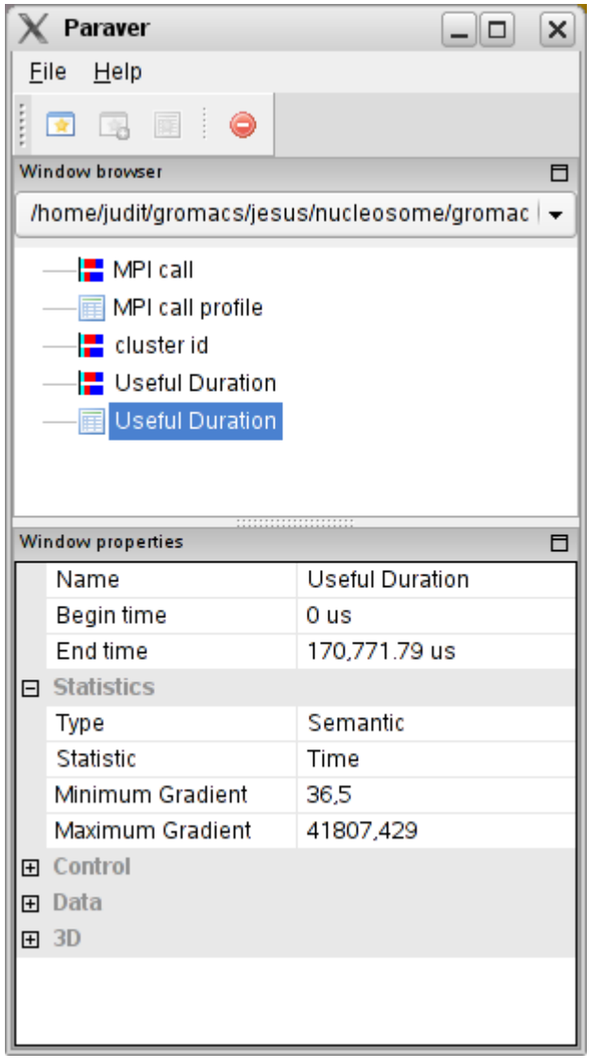
TRACE.sym

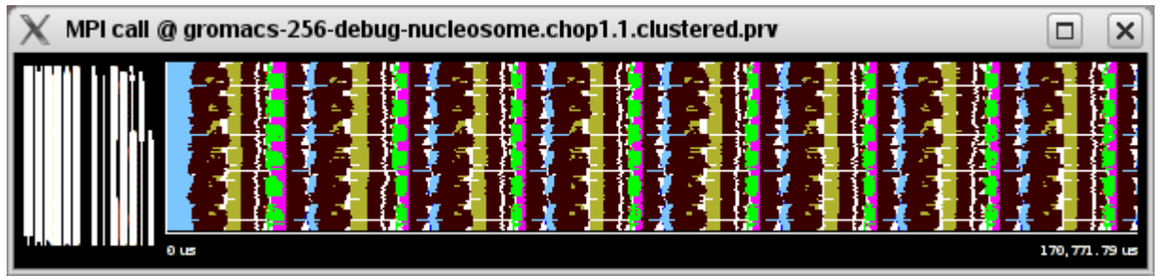
Examples of MN set-up available at /gpfs/apps/CEPBATOOLS/tracing-setup

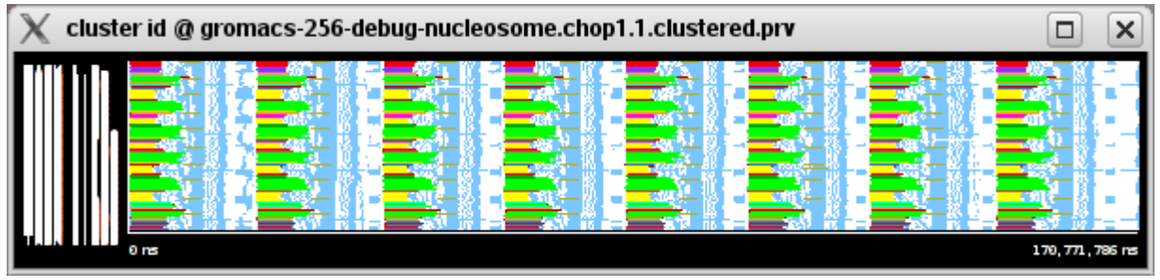
Paraver - Visualization and analysis

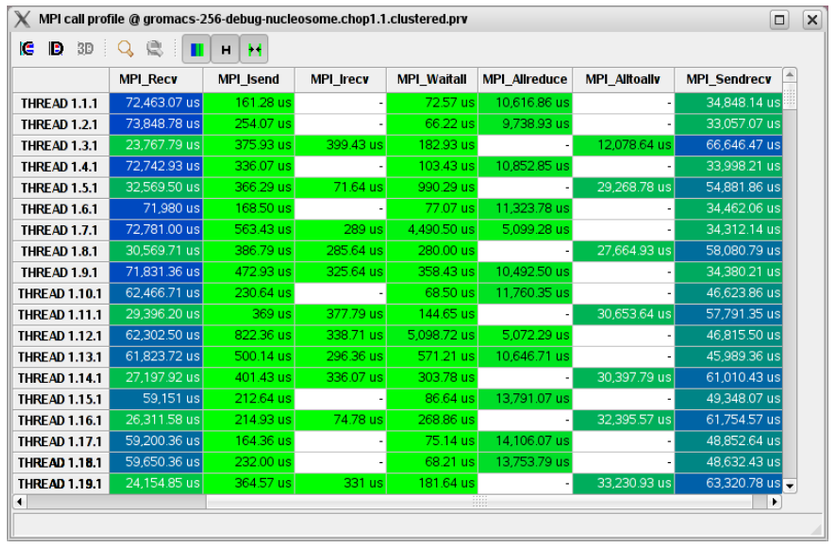
New version of Paraver

- GUI based on wxWidgets – Linux and Windows

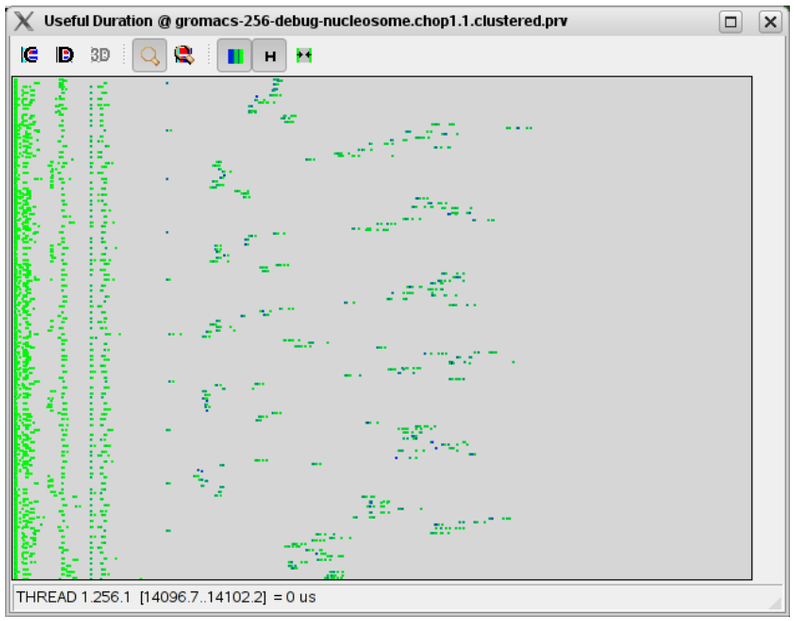








	MPI_Recv	MPI_Isend	MPI_Irecv	MPI_Waitall	MPI_Allreduce	MPI_Alltoallv	MPI_Sendrecv
THREAD 1.1.1	72,463.07 us	161.28 us	-	72.57 us	10,616.86 us	-	34,848.14 us
THREAD 1.2.1	73,848.78 us	254.07 us	-	66.22 us	9,738.93 us	-	33,057.07 us
THREAD 1.3.1	23,767.79 us	375.93 us	399.43 us	182.93 us	-	12,078.64 us	66,646.47 us
THREAD 1.4.1	72,742.93 us	336.07 us	-	103.43 us	10,852.85 us	-	33,998.21 us
THREAD 1.5.1	32,569.50 us	366.29 us	71.64 us	990.29 us	-	29,268.78 us	54,881.86 us
THREAD 1.6.1	71,980 us	168.50 us	-	77.07 us	11,323.78 us	-	34,462.06 us
THREAD 1.7.1	72,781.00 us	563.43 us	289 us	4,490.50 us	5,099.28 us	-	34,312.14 us
THREAD 1.8.1	30,569.71 us	386.79 us	285.64 us	280.00 us	-	27,664.93 us	58,080.79 us
THREAD 1.9.1	71,831.36 us	472.93 us	325.64 us	358.43 us	10,492.50 us	-	34,380.21 us
THREAD 1.10.1	62,466.71 us	230.64 us	-	68.50 us	11,760.35 us	-	46,623.86 us
THREAD 1.11.1	29,396.20 us	369 us	377.79 us	144.65 us	-	30,653.64 us	57,791.35 us
THREAD 1.12.1	62,302.50 us	822.36 us	338.71 us	5,098.72 us	5,072.29 us	-	46,815.50 us
THREAD 1.13.1	61,823.72 us	500.14 us	296.36 us	571.21 us	10,646.71 us	-	45,989.36 us
THREAD 1.14.1	27,197.92 us	401.43 us	336.07 us	303.78 us	-	30,397.79 us	61,010.43 us
THREAD 1.15.1	59,151 us	212.64 us	-	86.64 us	13,791.07 us	-	49,348.07 us
THREAD 1.16.1	26,311.58 us	214.93 us	74.78 us	268.86 us	-	32,395.57 us	61,754.57 us
THREAD 1.17.1	59,200.36 us	164.36 us	-	75.14 us	14,106.07 us	-	48,852.64 us
THREAD 1.18.1	59,650.36 us	232.00 us	-	68.21 us	13,753.79 us	-	48,632.43 us
THREAD 1.19.1	24,154.85 us	364.57 us	331 us	181.64 us	-	33,230.93 us	63,320.78 us

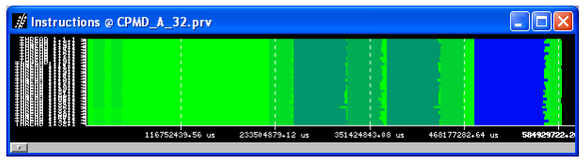


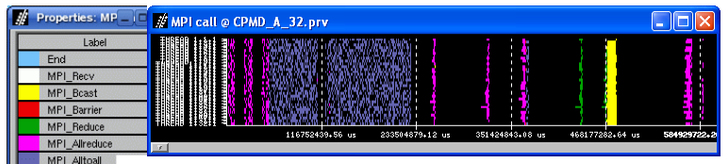
Views: Timelines

- From raw events → Piece-wise constant functions of time → plots / colors

- Basic metrics

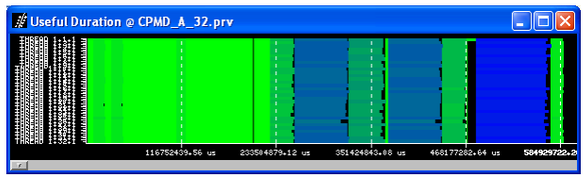
Instructions





MPI calls

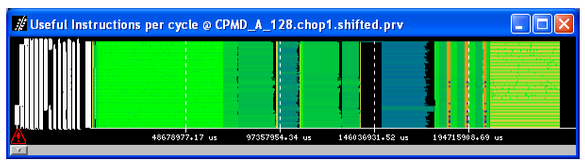
Useful duration



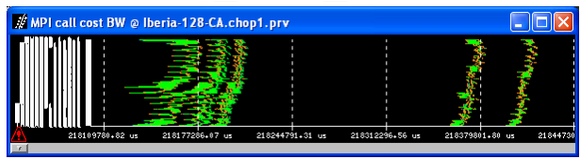
Color encoding

- Derived metrics

$$useful\_IPC = \frac{\#instr}{\#cycles} * useful$$



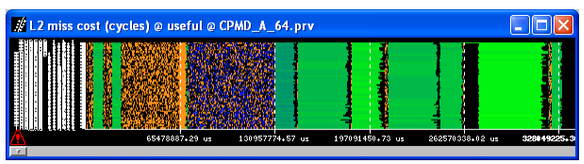
$$MPI\_call\_Cost = \frac{MPI\_call\_duration}{\#bytes}$$



$$preempted\_time = elapsed - \frac{cycles}{clock\_freq}$$

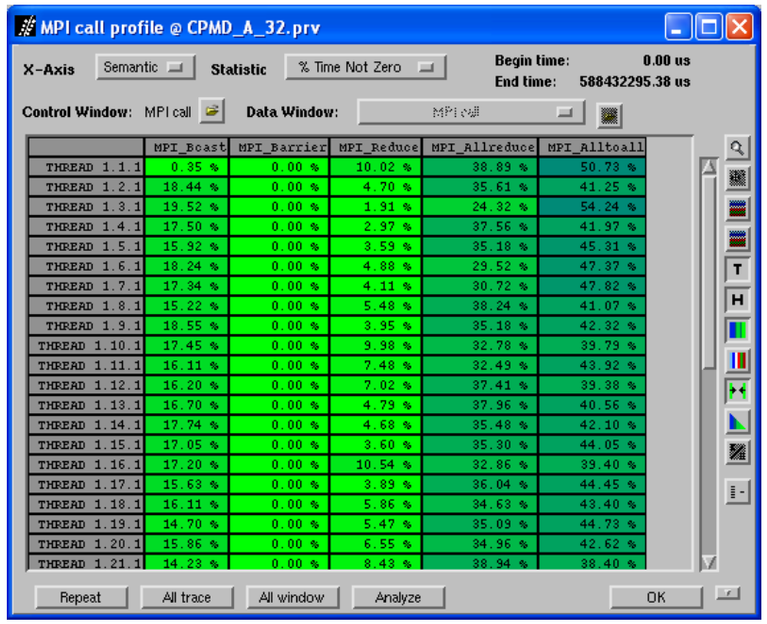
- Models

$$L2\_miss\_latency = \frac{\#cycles - \#instr / idealIPC}{\#L2misses}$$

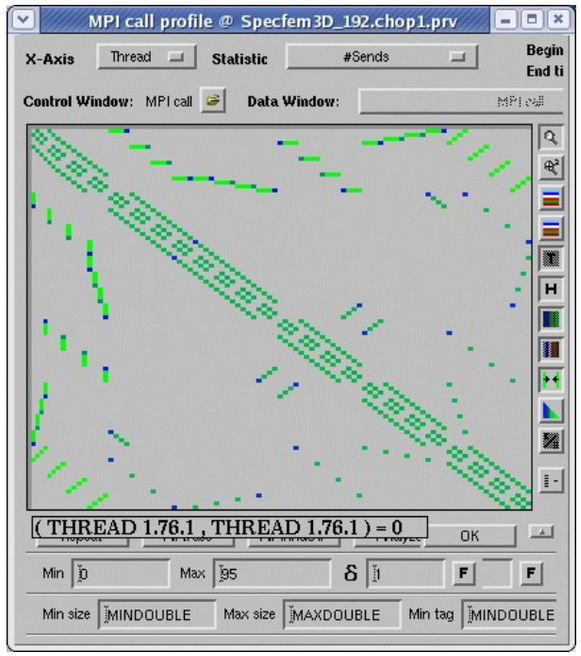


Views: Statistics

- 2D
  - Profiles
  - Histograms
  - Correlations
  - Communication Patterns

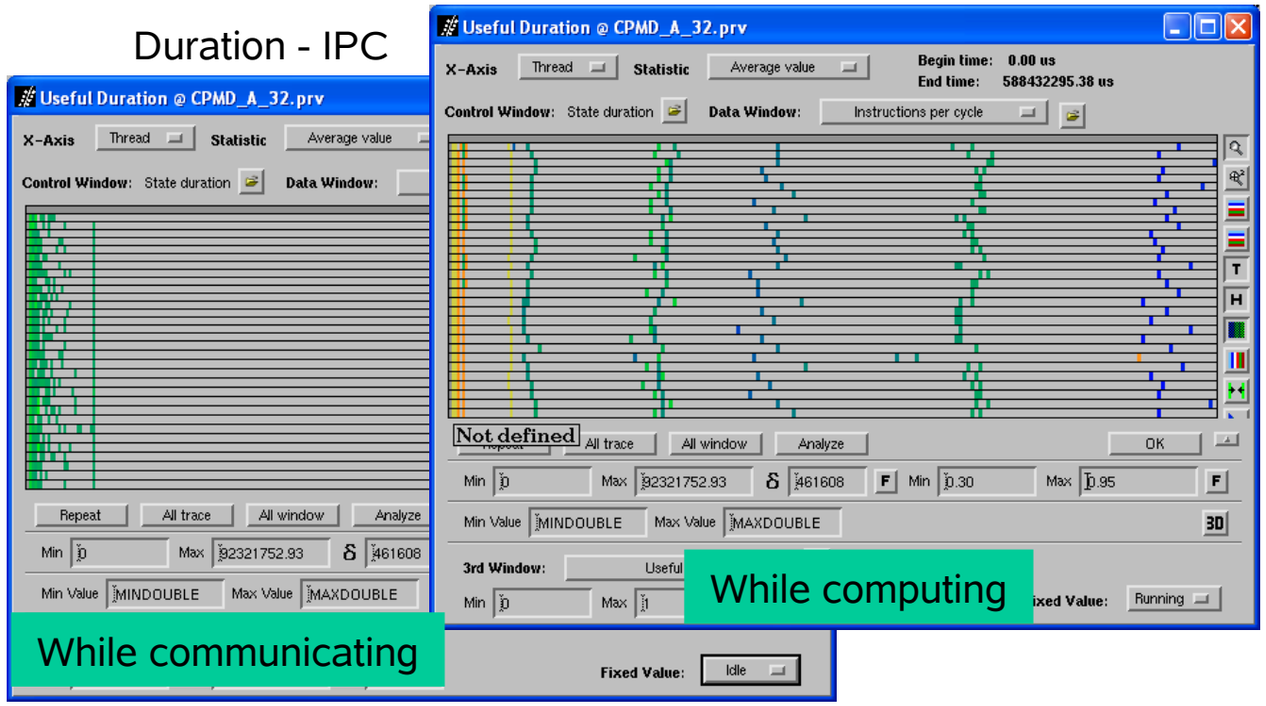


MPI calls profile



- 3D

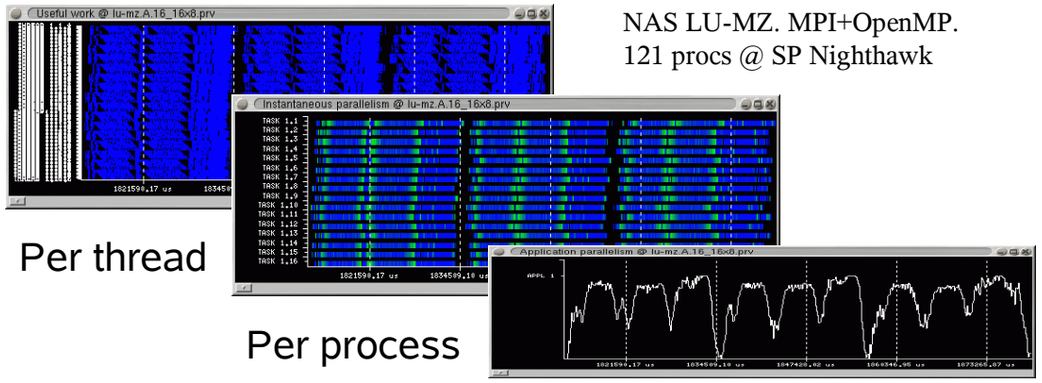
Duration - IPC



Scalability

Scalability of Presentation

- Aggregation
  - Functional rather than scalability motivation



NAS LU-MZ. MPI+OpenMP.
121 procs @ SP Nighthawk

Per thread

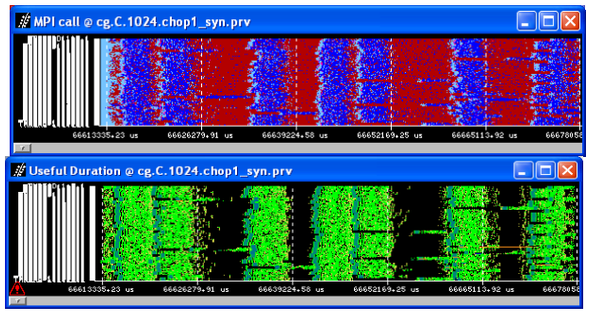
Per process

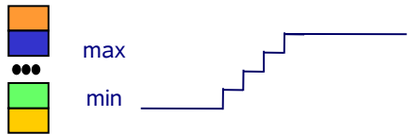
Per application

- Display
  - Non linear render
    - Value for pixel
    - Colors

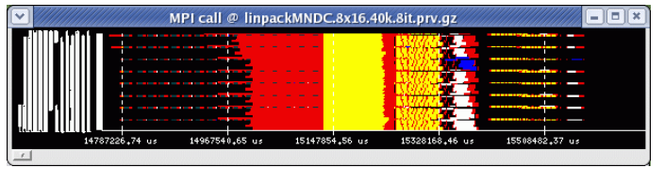
Random

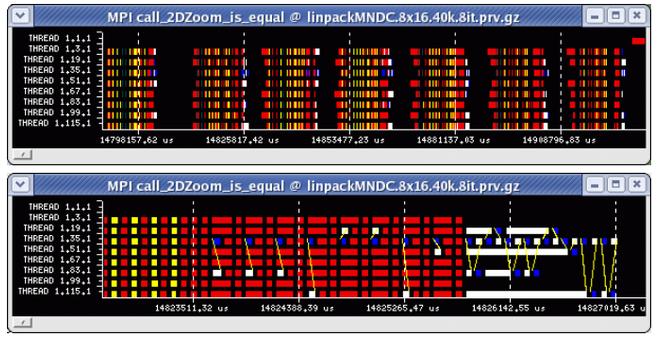
Max





- Objects
  - Any subset





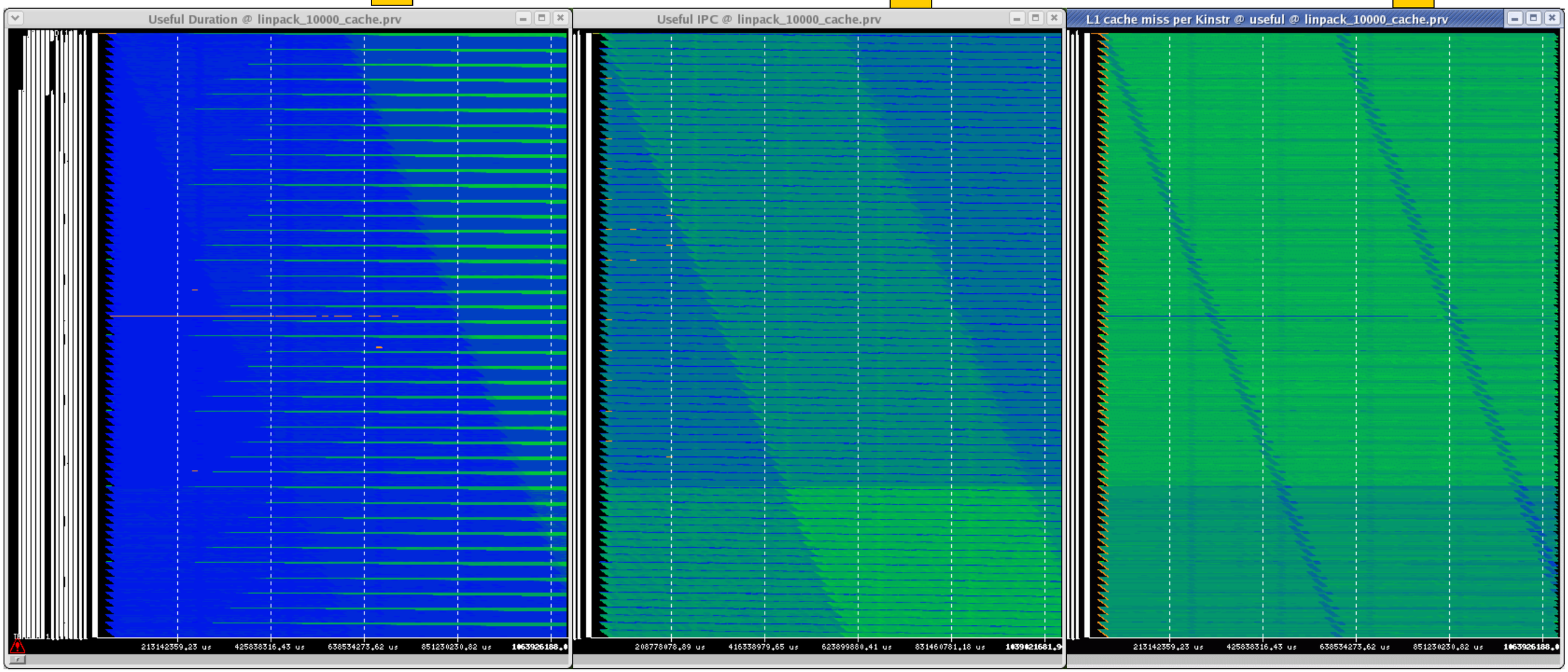
Scalability of Presentation

- Linpack @ Marenostrum: 10k cores x 1700 s

Dgemm duration: 11.8 s ••• 10 s

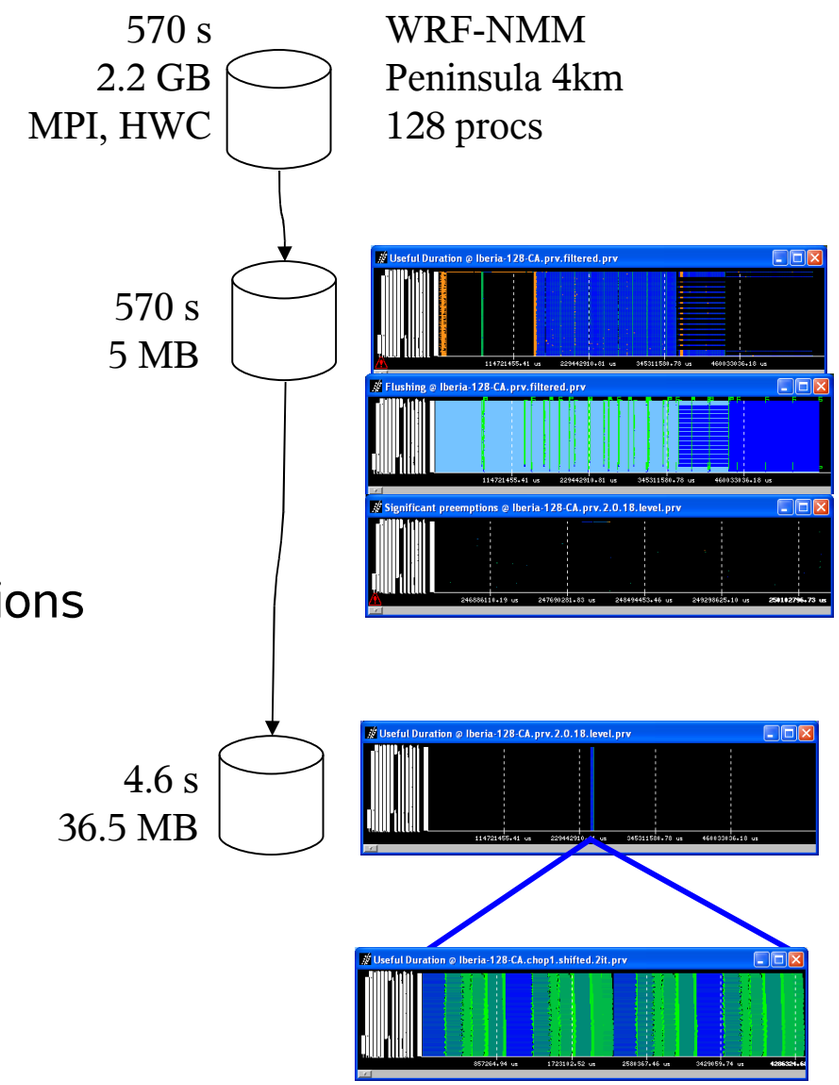
Dgemm IPC: 2.95 ••• 2.85

Dgemm L1 miss ratio: 0.8 ••• 0.7



Scalability: Data reduction through periodicity analysis

- Data handling/summarization capability
  - Software counters, filtering and cutting
  - Supported by GUI.
- Automatizable through signal processing techniques:
  - Mathematical morphology to clean up perturbed regions
  - Wavelet transform to identify coarse regions
  - Spectral analysis for detailed periodic pattern
- Algorithms currently applied to traces
  - to be ported to run time



- Similar to manual structure identification

Scaling by selectively emitting trace data

- Gadget: 15.5 MB (3 MB compressed) → Init, 3 iters (10.6 s); termination

2048 procs

Useful duration

% MPI time

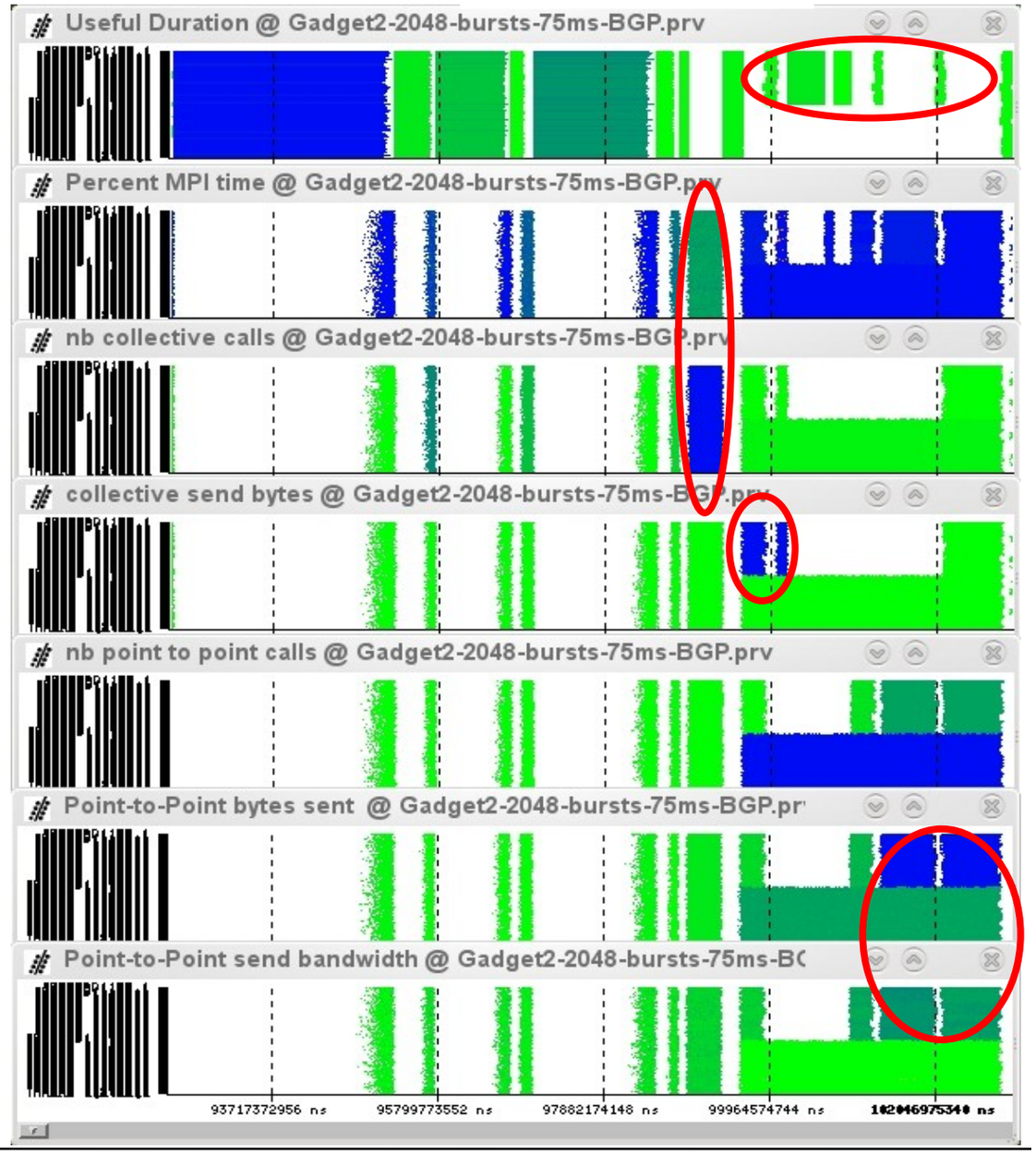
collectives

Collective bytes

p2p

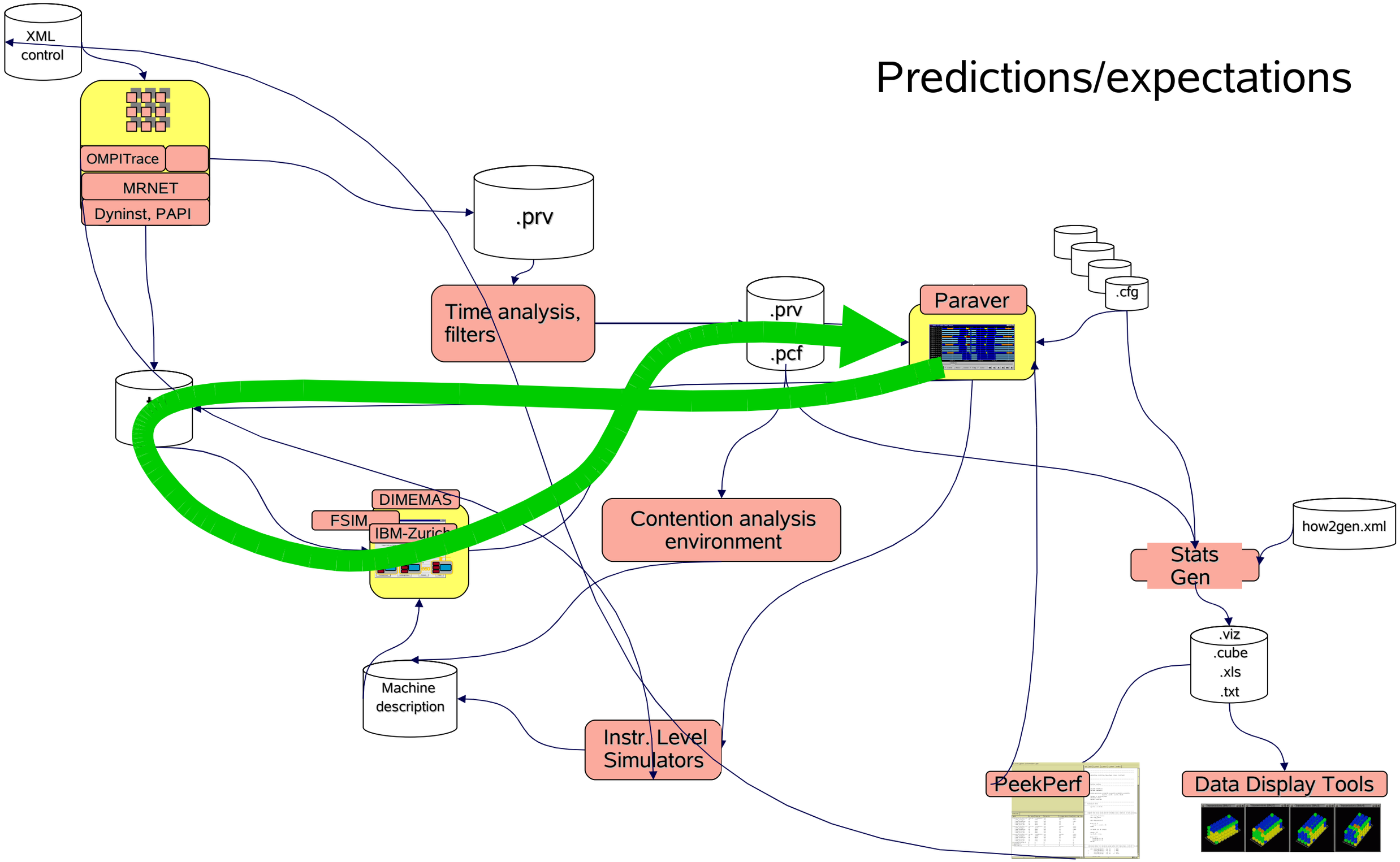
p2p bytes

p2p BW



Dimemas: Models and predictions

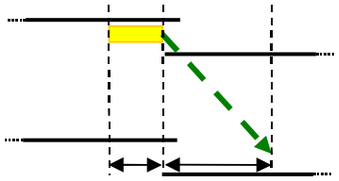
CEPBA-Tools Environment

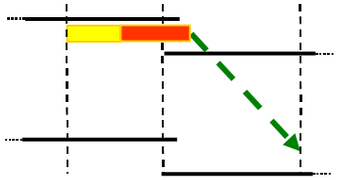


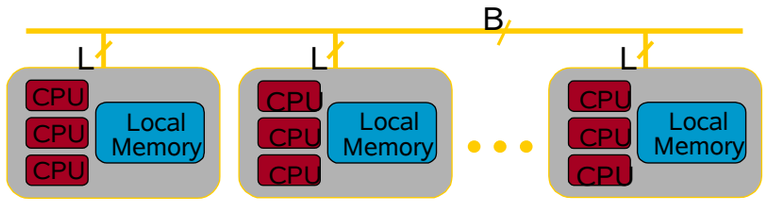
Dimemas: Coarse grain, Trace driven simulation

- Simulation: Highly non linear model
  - Linear components
    - Point to point communication
    - Sequential processor performance
      - Global CPU speed
      - Per block/subroutine
  - Non linear components
    - Synchronization semantics
      - Blocking receives
      - Rendezvous
    - Resource contention
      - CPU
      - Communication subsystem
        - links (half/full duplex), busses

$$T = \frac{MessageSize}{BW} + L$$

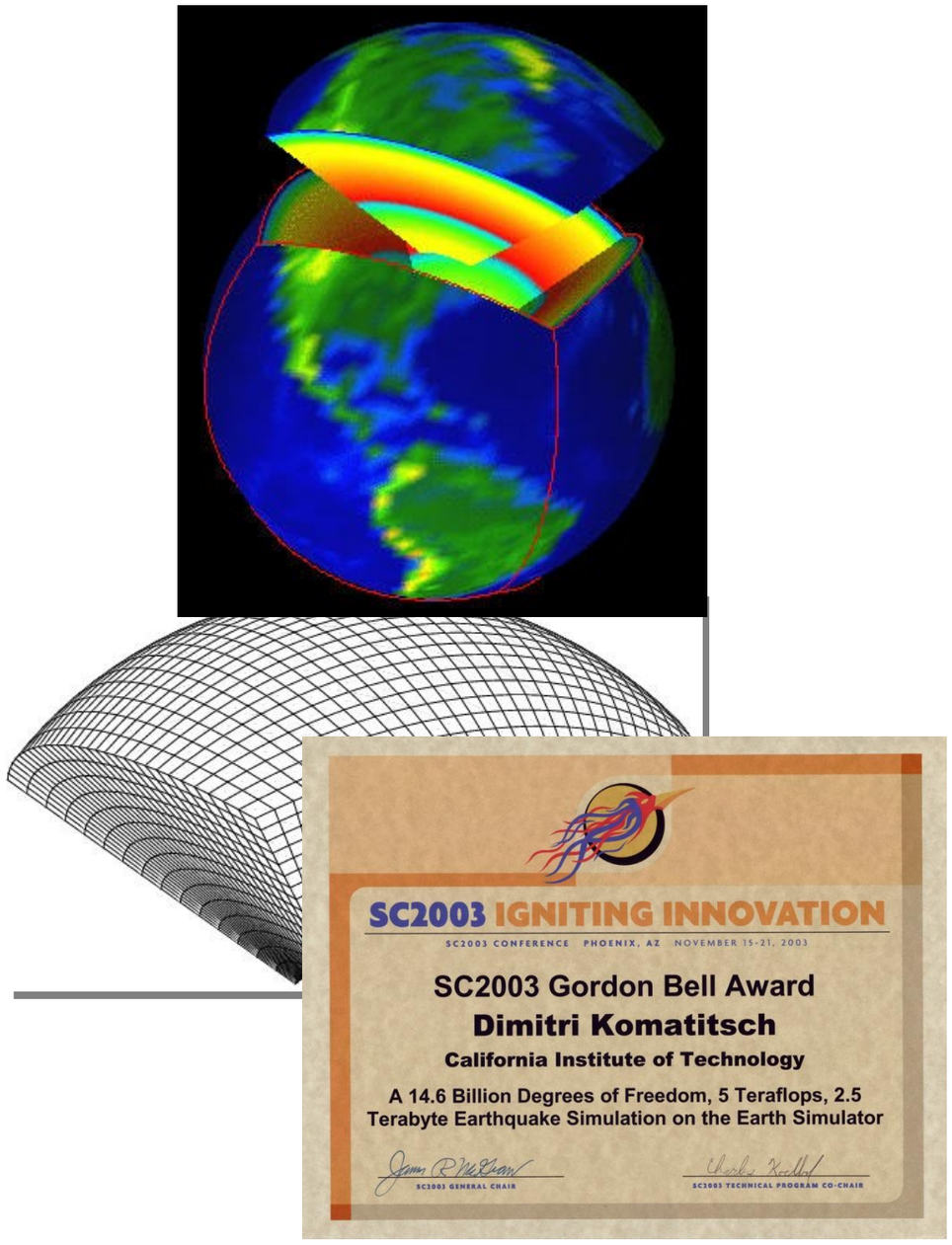






Example: Specfem3D

- Should I introduce asynchronous communication?



Courtesy Dimitri Komatitsch

Real

ideal

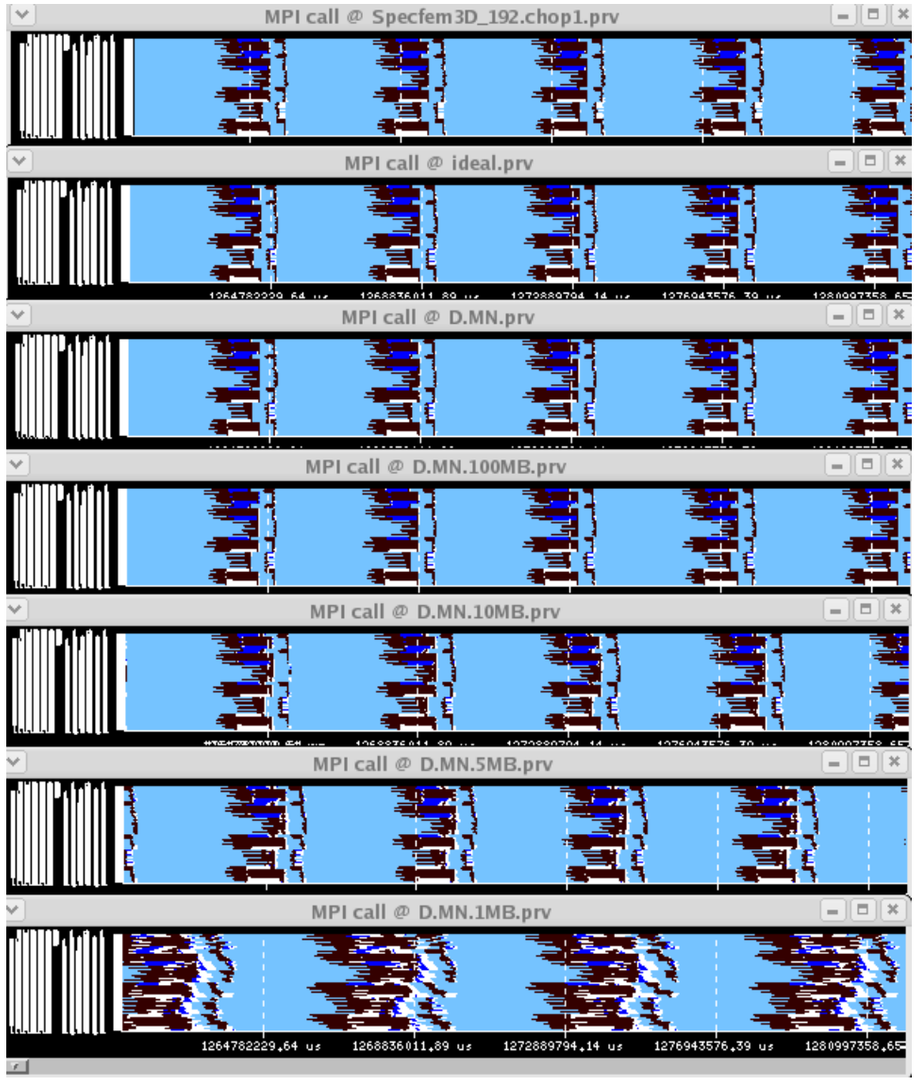
NM prediction

Prediction 100MB/s

Prediction 10MB/s

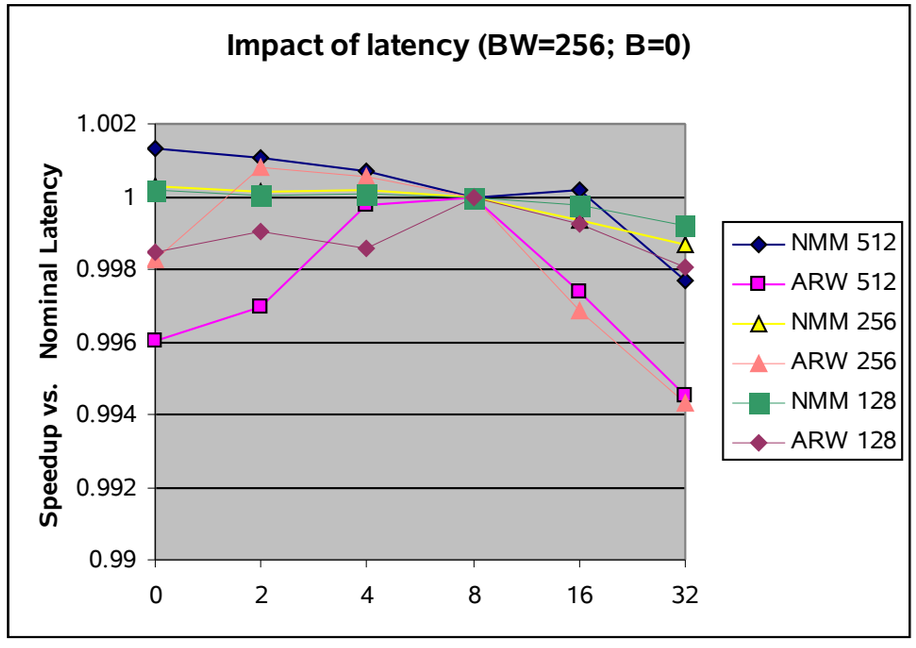
Prediction 5MB/s

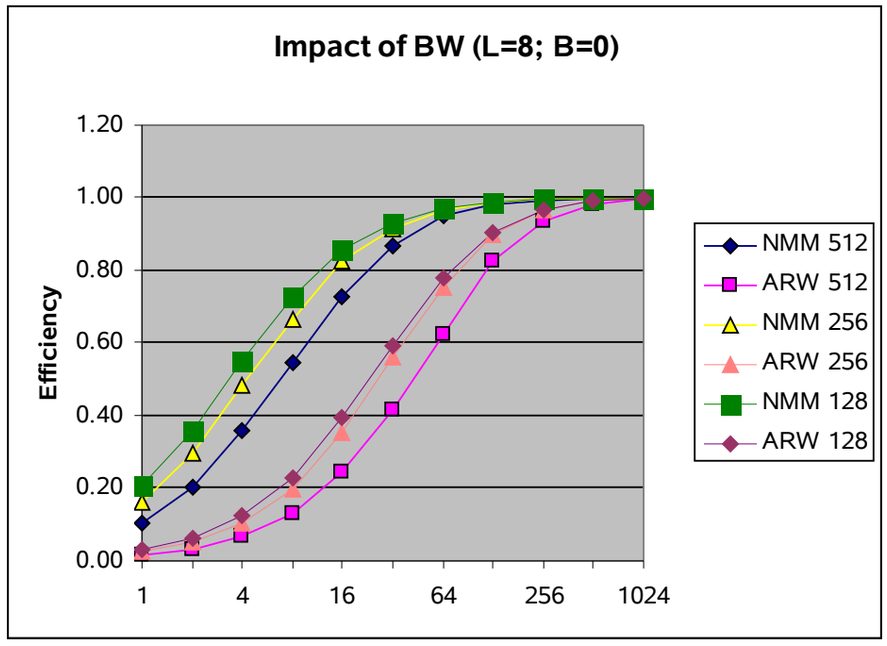
Prediction 1MB/s

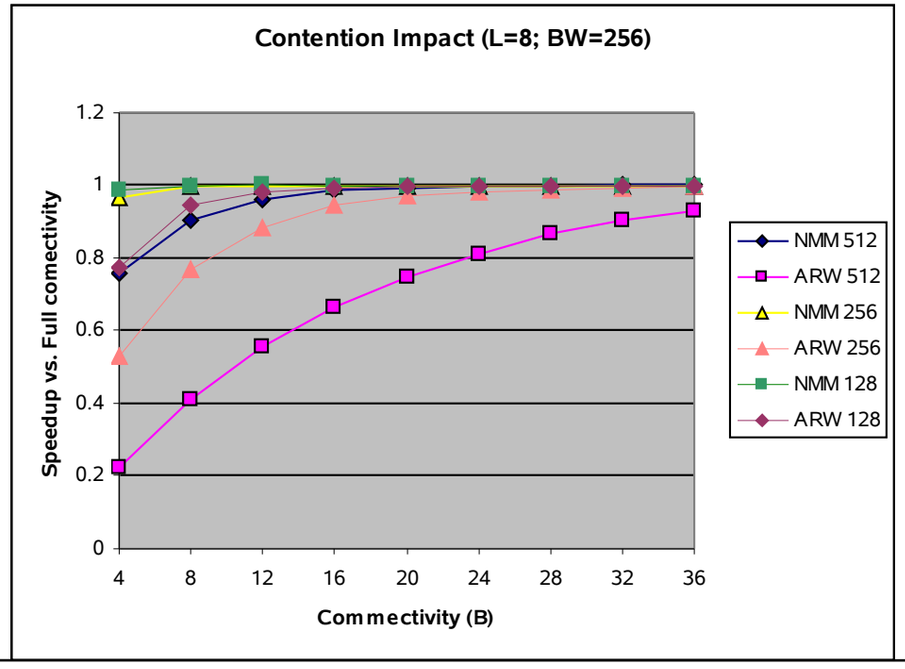


Example: Network sensitivity

- Simulations with 4 processes per node
- NMM Iberia 4Km
  - Not sensitive to Latency
  - 512 sensitive to contention?
  - 256 MB/s OK
- ARW Iberia 4 Km
  - Not sensitive to Latency
  - sensitive to contention
  - Need 1GB/s



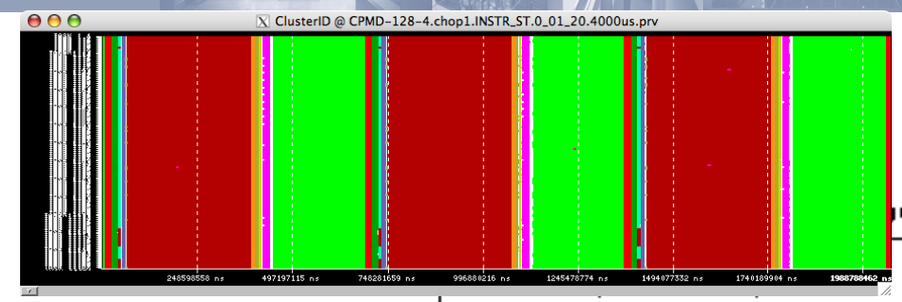




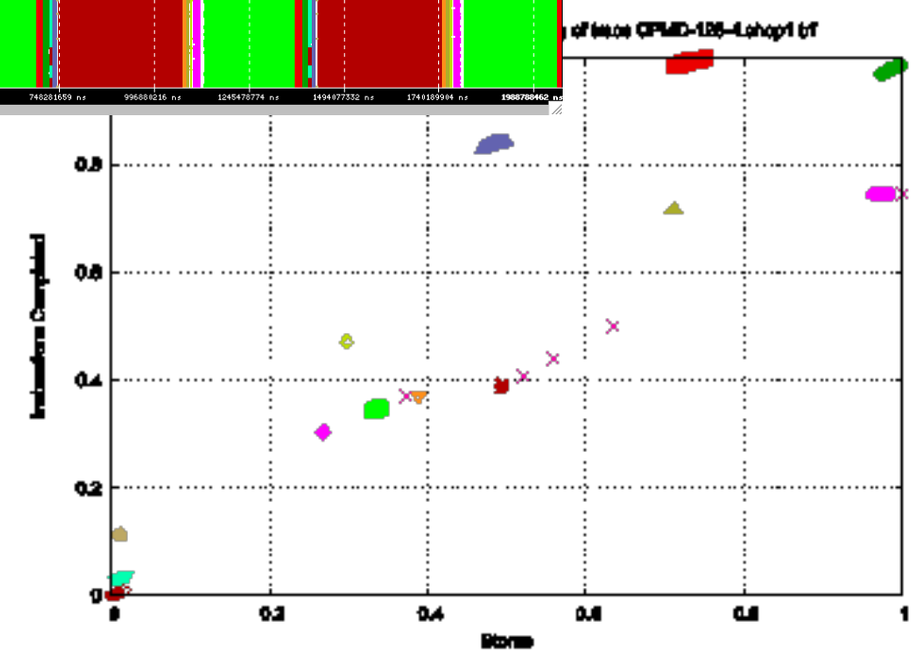
Clustering – Detecting structure

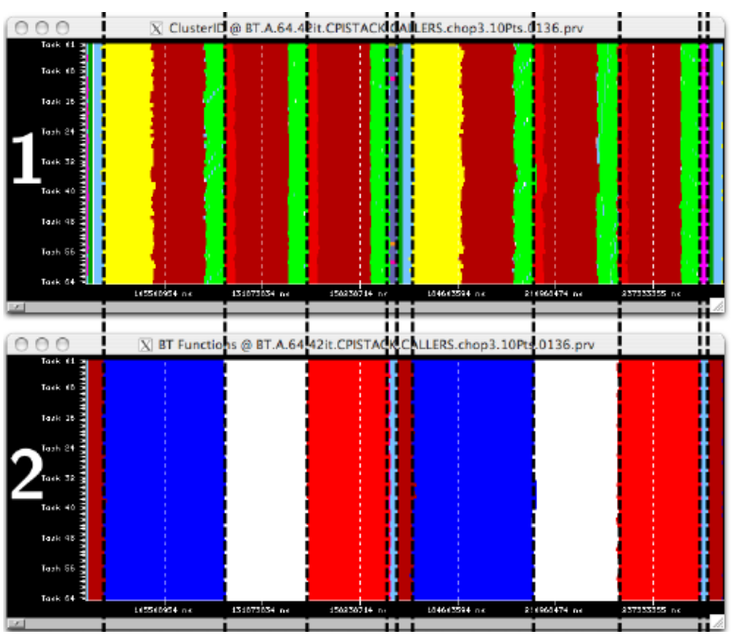
Clustering

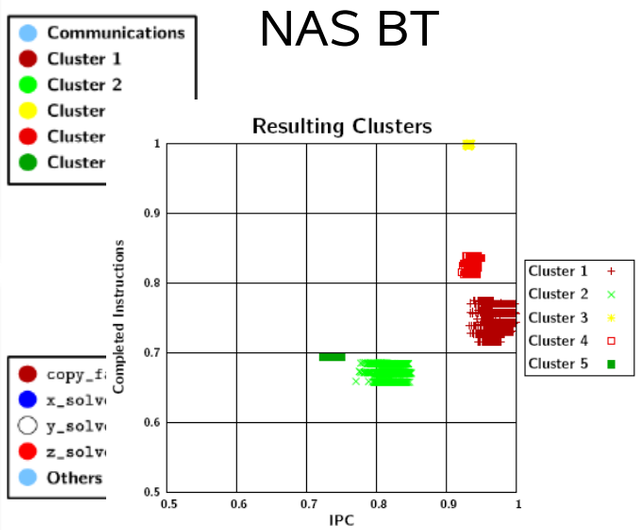
- Useful for
  - Identifying and highlighting structure
    - Cluster information injected in trace
    - Phases within routines
    - Different routines may have similar behavior
  - Compact trace encoding
  - Input to time analysis



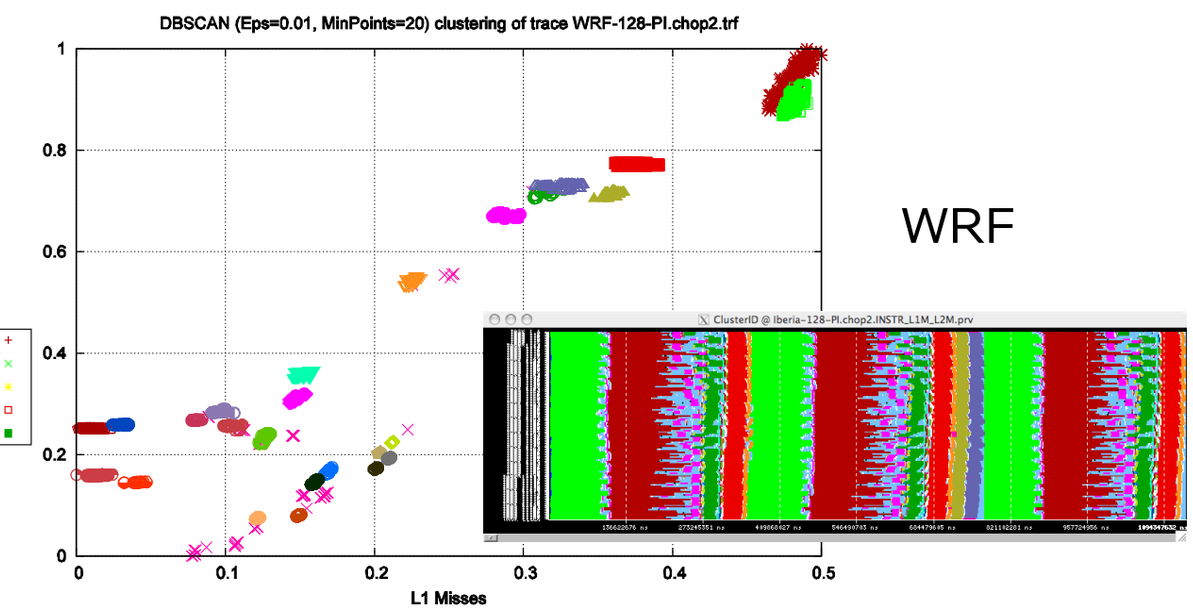
CPMD







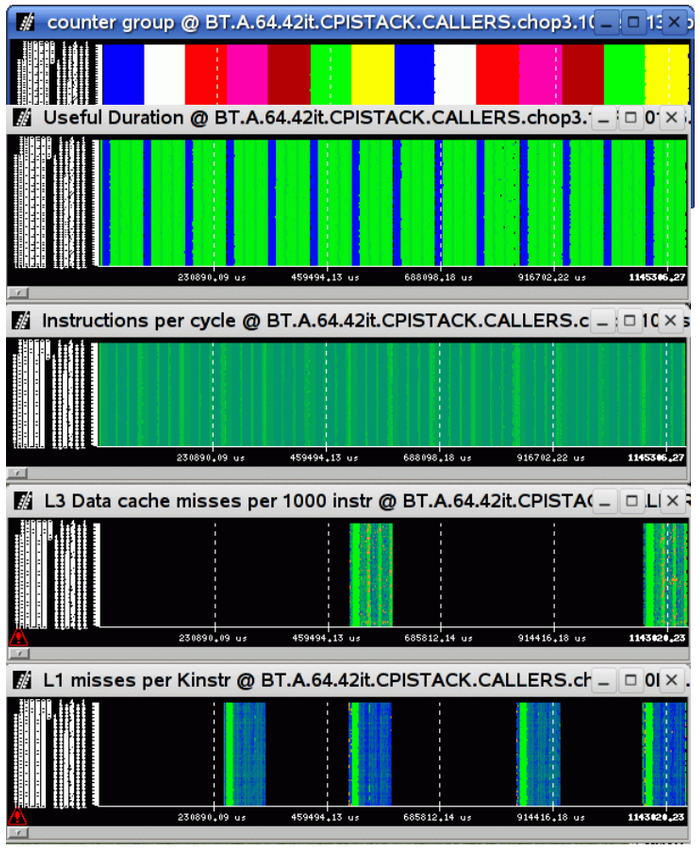
NAS BT



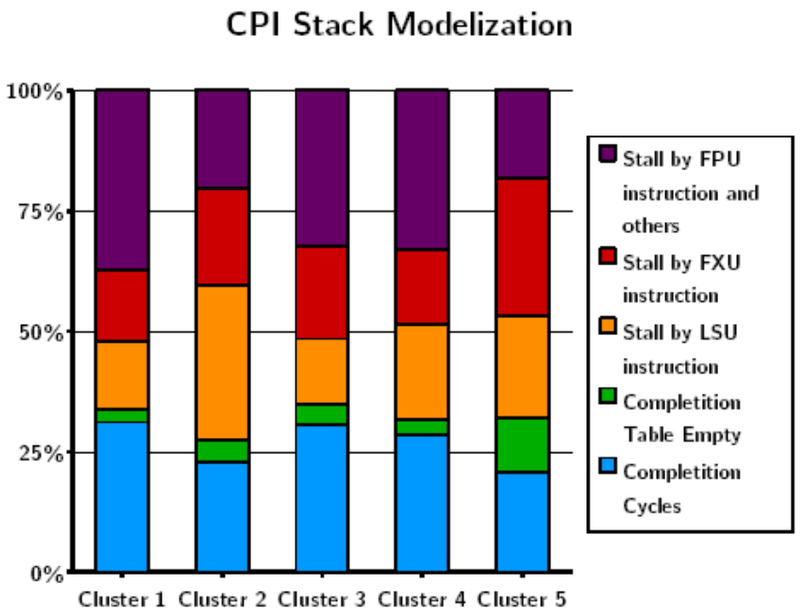
WRF

Clustering

- Useful for
  - Precise projection of hardware counters
    - Statistics
    - CPI stack model

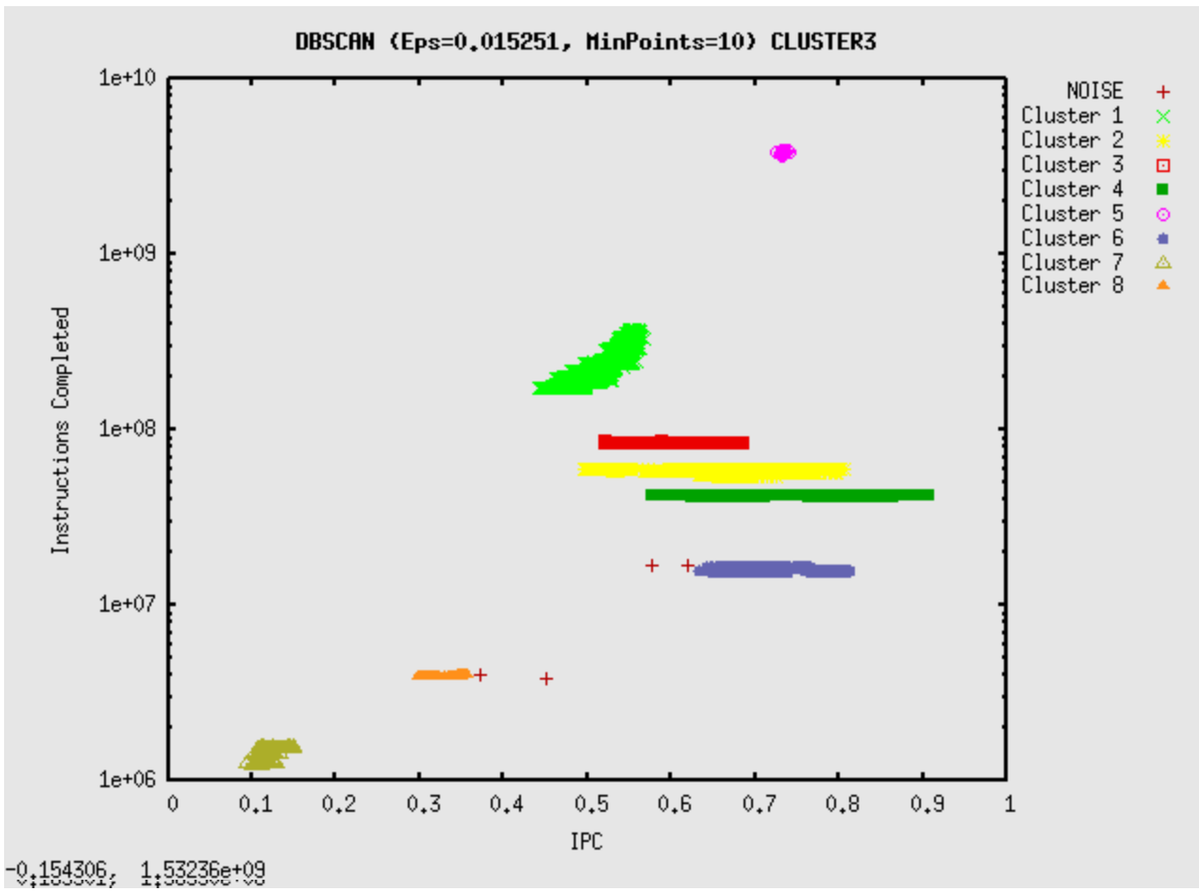


Cluster	1	2	3	4	5
%Time	54.88	17.96	16.90	6.44	1.42
Avg. Burst Dur. (ms)	1.02	0.78	13.14	2.50	1.11
IPC	1.02	0.65	0.89	0.91	0.53
MIPS	2231.8	1423.3	1966.5	2001.8	1163.0
MFLOPS	339.2	46.3	191.6	269.2	23.6
L1M/KInstr	0.92	1.53	1.19	1.17	2.88
L2M/KInstr	0.06	1.26	0.06	0.35	0.21
Mem.BW (MB/s)	16.79	218.47	13.87	85.77	29.76

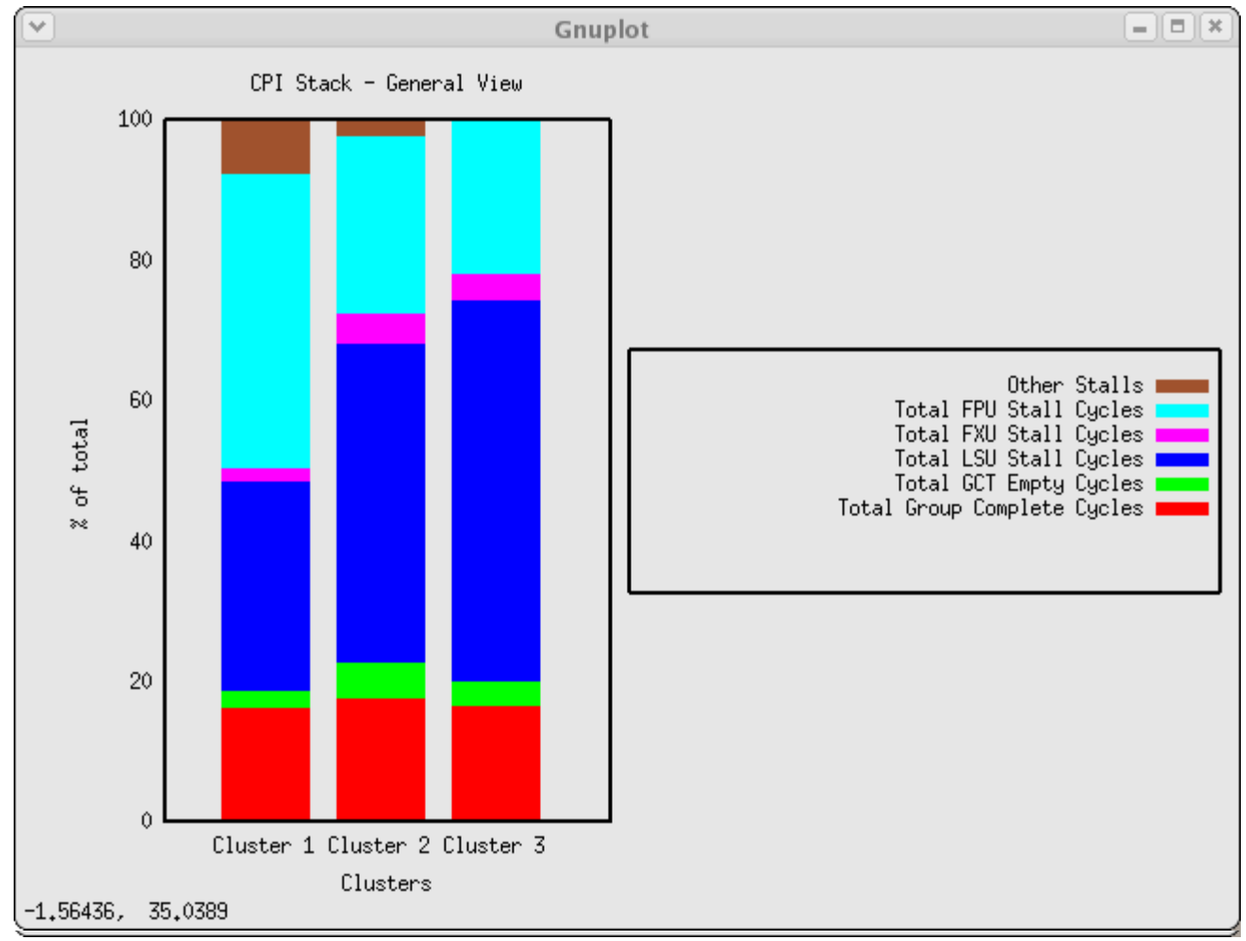


MRNet integration: Scalable online analysis

Analysis at minute 30



Clusters distribution



CPI STACK model (generic)

Sampling – Adding details

Sampling

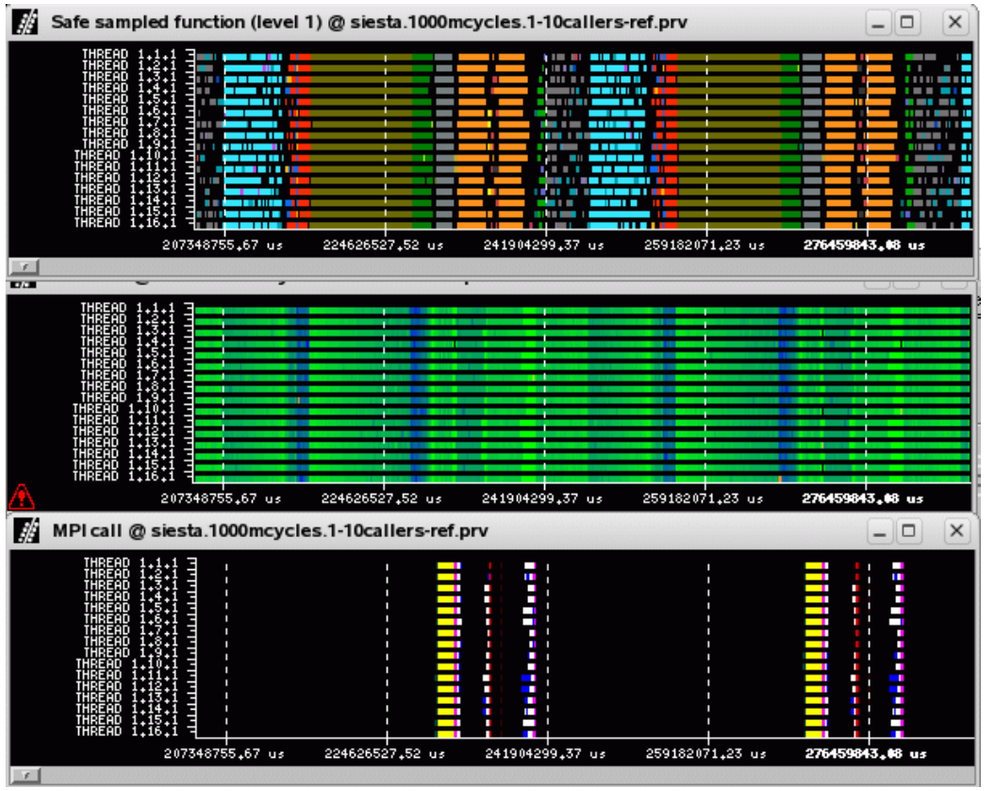
- Adding sampling information on the tracefile
  - based on the overflow mechanism offered by PAPI
    - period = f (cycles / instructions / cache misses...)
  - sampled information:
    - call stack – as reference to source
    - other hardware counters (not sampled)

- Useful to identify
  - Timeline profile
  - Application structure

Safe sampled functions

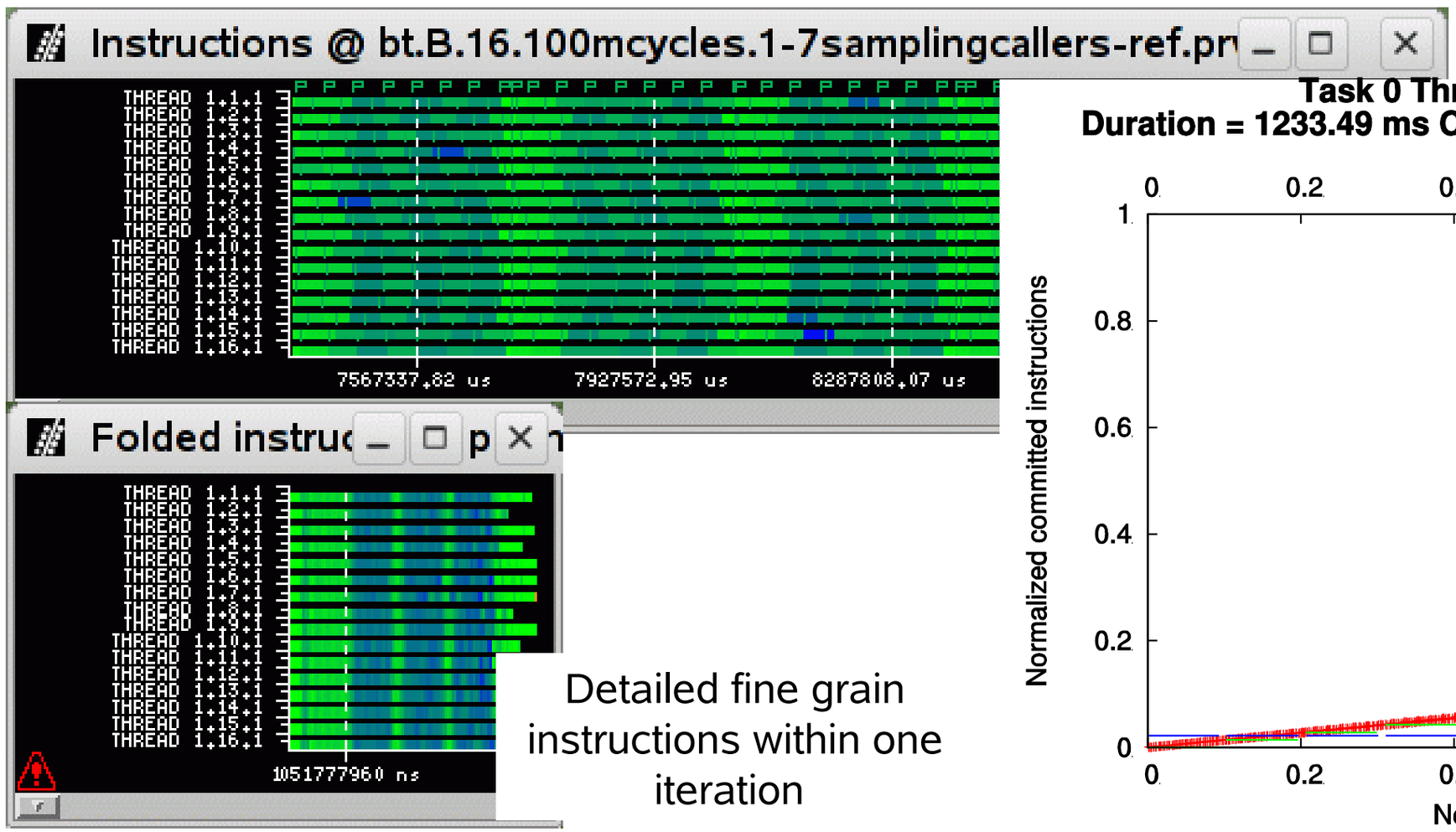
MFLOPS at each sampled interval

Instrumented MPI calls



Sampling

- Low sampling frequency (<< Nyquist)
  - Folding mechanism to achieve high precision
  - Assumes stationary periodic behavior
    - event at start of iteration



Detailed fine grain instructions within one iteration