JÜLICH
FORSCHUNGSZENTRUM

BSC

German Research School
for Simulation Sciences

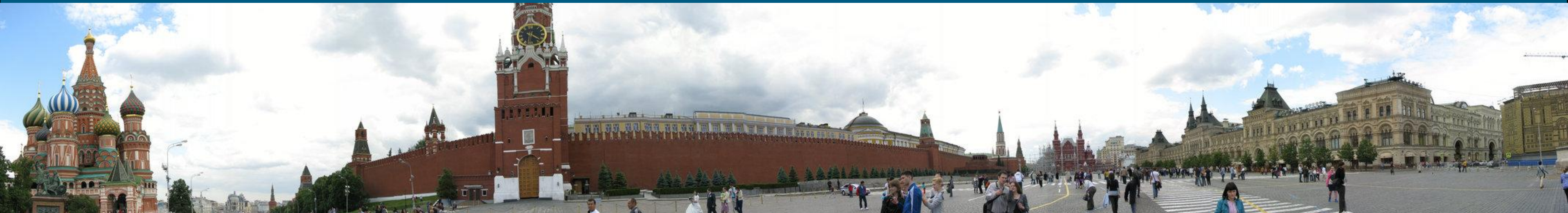
ROGUE WAVE
SOFTWARE

ZIH

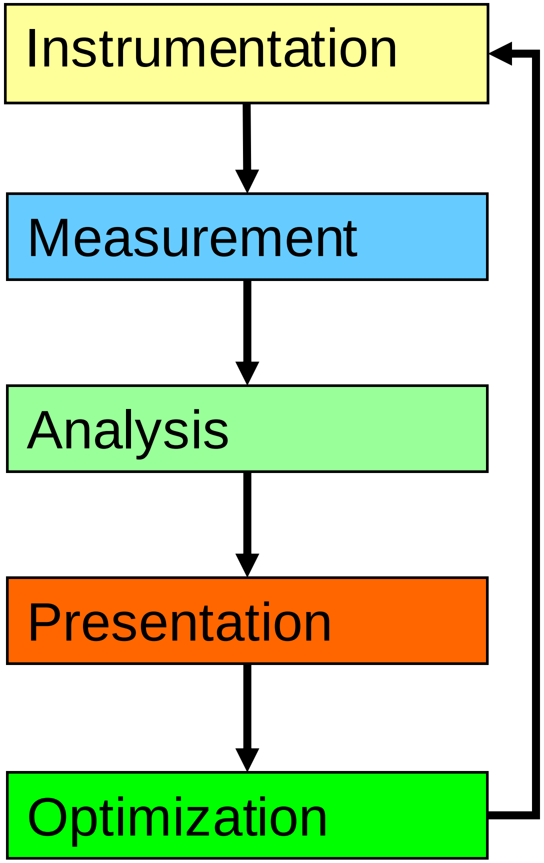
PLATFORMS

Quick Introduction to Parallel Performance Analysis

Bernd Mohr, Jülich Supercomputing Centre



Performance Measurement Cycle



- Insertion of extra code (probes, hooks) into application
- Collection of data relevant to performance analysis
- Calculation of metrics, identification of performance problems
- Transformation of the results into a representation that can be easily understood by a human user
- Elimination of performance problems

Tool Support

Performance Measurement

- **Two dimensions**
  - **When** performance measurement is triggered
    - **Externally** (asynchronous) ⇨ indirect measurement
      - Sampling
        - Timer interrupt
        - Hardware counters overflow
    - **Internally** (synchronous) ⇨ direct measurement
      - Code instrumentation
        - Automatic or manual instrumentation
  - **How** performance data is recorded
    - **Profile** ::= Summation of events over time
      - run time summarization (functions, call sites, loops, …)
    - **Trace file** ::= Sequence of events over time

Measurement Methods: Profiling I

- Recording of **aggregated information**
    - Time
    - Counts
        - Calls
        - Hardware counters
- **about program and system entities**
    - Functions, call sites, loops, basic blocks, …
    - Processes, threads

- Result presentation as
    - Histograms, pie charts, …
    - Tables

Measurement Methods: Tracing

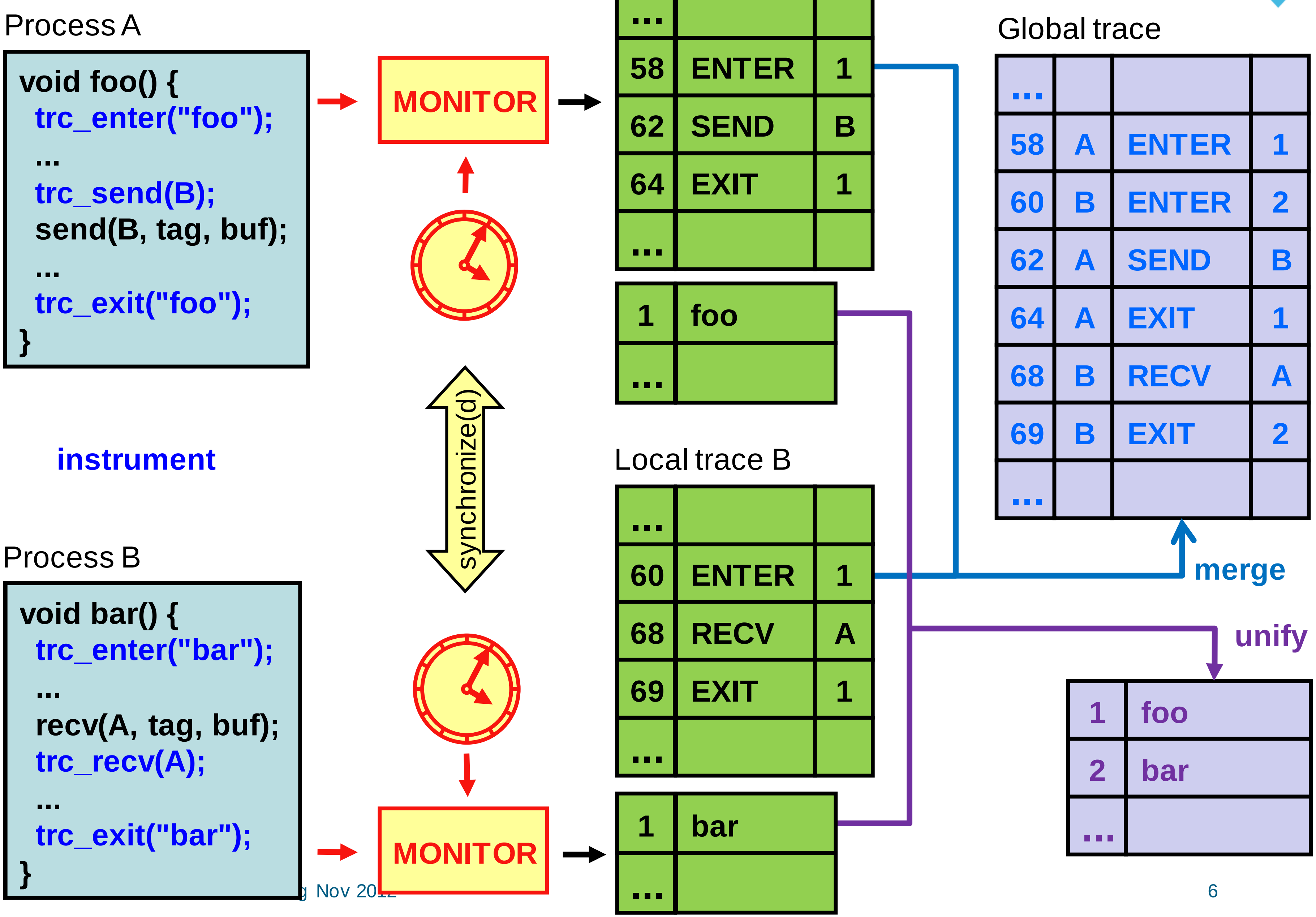
- Recording **information about** significant points (**events**) during execution of the program
  - Enter/leave a code region (function, loop, …)
  - Send/receive a message ...
- Save information in **event record**
  - Timestamp, location ID, event type
  - plus event specific information
- **Event trace** := stream of event records sorted by time

- Can be used to reconstruct the **dynamic behavior**

  ⇨ Abstract execution model on level of defined events

- Result presentation as **time line diagrams**

Event tracing



Process A

```
void foo() {
  trc_enter("foo");
  ...
  trc_send(B);
  send(B, tag, buf);
  ...
  trc_exit("foo");
}
```

instrument

Process B

```
void bar() {
  trc_enter("bar");
  ...
  recv(A, tag, buf);
  trc_recv(A);
  ...
  trc_exit("bar");
}
```

MONITOR

synchronize(d)

MONITOR

Local trace A

...		
58	ENTER	1
62	SEND	B
64	EXIT	1
...		

1	foo
...	

Local trace B

...		
60	ENTER	1
68	RECV	A
69	EXIT	1
...		

1	bar
...	

Global trace

...			
58	A	ENTER	1
60	B	ENTER	2
62	A	SEND	B
64	A	EXIT	1
68	B	RECV	A
69	B	EXIT	2
...			

merge

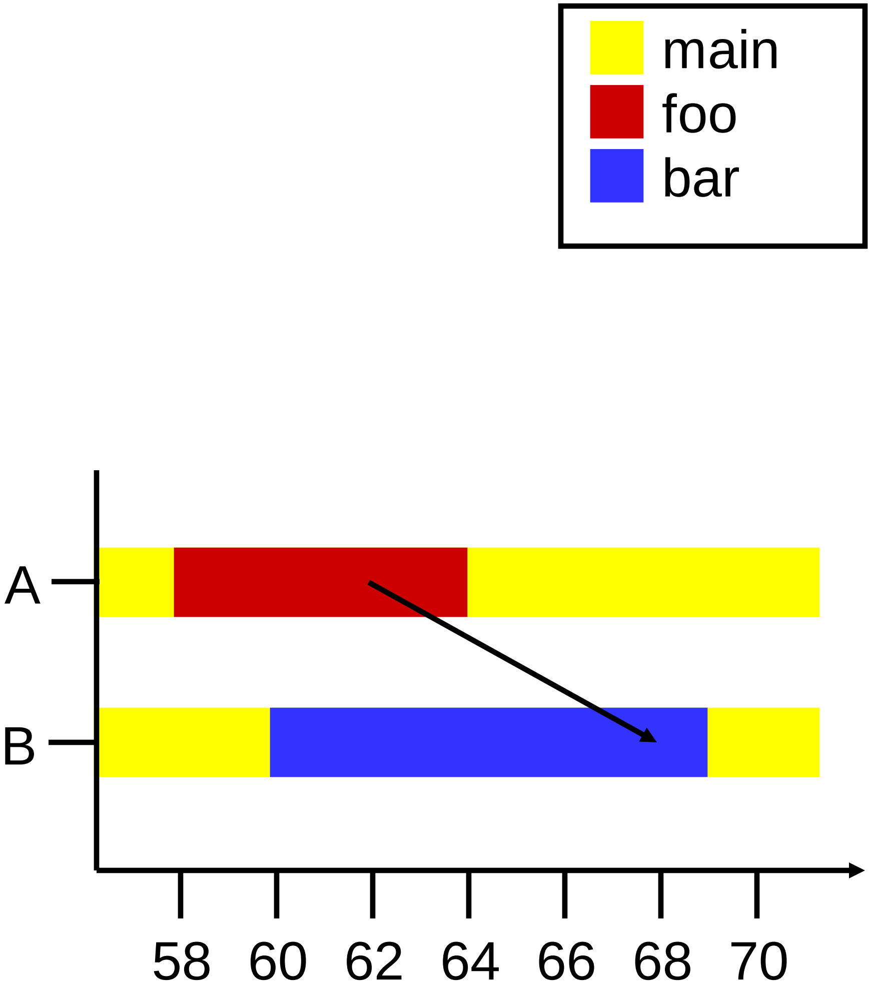
unify

1	foo
2	bar
...	

Event Tracing: “Timeline” Visualization

1	foo
2	bar
3	…

…			
58	A	ENTER	1
60	B	ENTER	2
62	A	SEND	B
64	A	EXIT	1
68	B	RECV	A
69	B	EXIT	2
…			



Questions?