Java Performance Measurement Framework An Overview

Lubomír Bulej

DISTRIBUTED SYSTEMS RESEARCH GROUP
http://dsrg.mff.cuni.cz

CHARLES UNIVERSITY PRAGUE
Faculty of Mathematics and Physics

Measuring performance

... in case of middleware/library

- Create benchmark application(s) for typical use cases, measure duration of actions and additional information such as e.g. amount of data sent over network, store the measured data

... in case of an application

- Decide what needs to be measured, find spots in the application where to measure it, modify the application to do the measurement, the rest as above...

Simplifying the measurement

Application specific tasks

- Define and ensure generation of performance events
- Define data to be collected when events occur

Common, technical tasks

- Collect and store data associated with the events

Simplification through separation of concerns

- Application instrumentation
- Measurement configuration
- Data collection and storage

Measurement framework responsibilities

Allow to define performance events

- Specific, through manual instrumentation
  - Counting the number of iterations in selected loops
- Generic, through automatic instrumentation
  - Method invocations on component interfaces

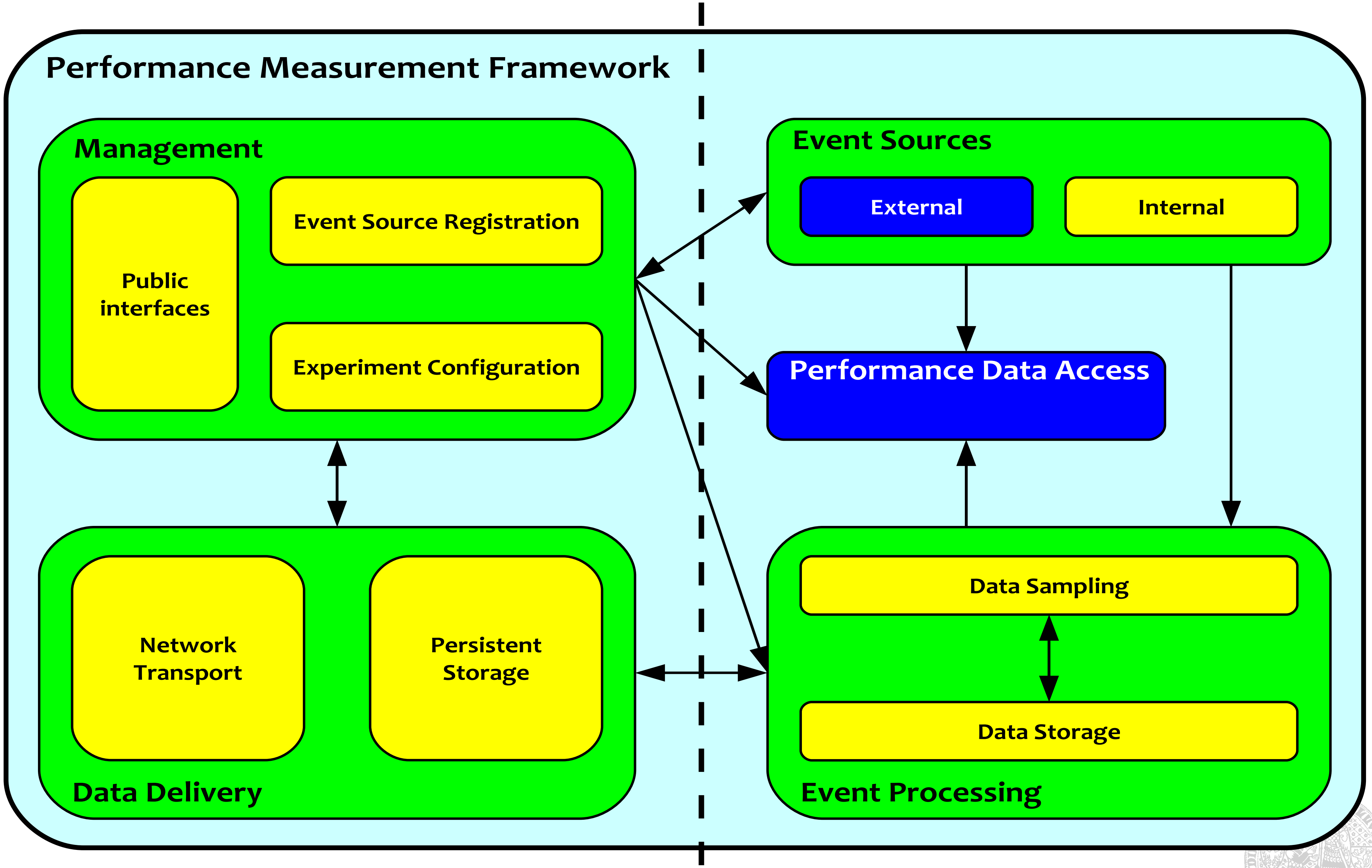
Allow to configure what data to collect

- Event-specific data
  - iteration counts, time stamps, sizes of arguments, ...
- Generic system-wide data
  - amount of data transferred over network, ...

Collect and store the data

- Without the user having to care about details

Framework architecture overview



Events & event sources

Event type

- Interface with methods for related events
  - e.g. enterMethod(), leaveMethod()

Event source

- Supports multiple event groups of (various) given types
  - e.g. single event source would correspond to a single component interface, the event source providing enterMethod() and leaveMethod() events for each of the interface methods
- Individual event groups can be enabled/disabled
  - i.e. for particular method
- Event sources can be enabled/disabled
  - e.g. to reduce overhead if no events are enabled

External event sources & event triggers

External event sources

- Client needs to implement event source interface and register instances with the framework
  - Implemented externally, typically provided by automatic instrumentation
  - Framework controls the event source

Event triggers

- Simplified internal implementation of event source interface, simple events only
  - Typically instantiated by manual instrumentation
  - Provides client with methods for generating events
  - Framework controls the (internal) event source

Performance data access

Provides generic access to...

- Time sources
  - e.g. CPU-based timers, HPET, RTC, gettimeofday() ...
- Performance data sources
  - e.g. Windows performance objects, Linux /proc, Solaris kstat
  - Performance counters/gauges represented as sensors

Measurement context

- Represents a set of sensors from which to get readings
- Preconfigured to avoid disruptive operations (e.g. memory allocation, opening files) during measurement
- Provides prepare(), sample() & decode() methods, data accessed in generic way through value handles (holders)

Event processing & data delivery

Event delegates

- Callback references used by event sources to notify framework about events
  - Assigned to event sources by framework
- Collects generic performance data if configured so
  - Uses a preconfigured measurement context associated with a particular event group
- Stores event-specific and generic performance data to a preallocated in-memory buffer

Data delivery

- Periodically (or on demand) collects data from in-memory buffers and sends them over network or writes them to file in a generic format

Development status

Essential features under development

- Working Java prototype expected by end of June 2009
  - Event sources management and control
  - Event delegates and triggers for basic event types
  - In-memory storage for event specific data
  - Dump data to files
- Automatic instrumentation of Itemis showcase

Non-essential features as needed

- Generic performance data access, can be scaled down to only provide access to specific (not all) system-wide data
- In-place aggregation, advanced delivery methods, flexible configuration not as important at the moment