UHH Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CAU Christian-Albrechts-Universität zu Kiel

HITeC

Lancaster University Leipzig

Kieker

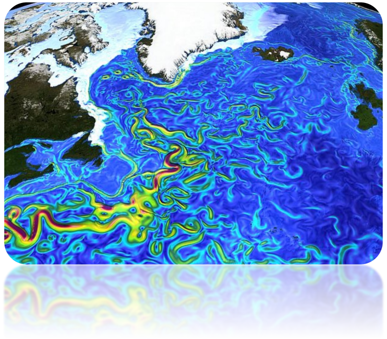
https://kieker-monitoring.net/

Update @ SSP 2022 – Nov 8, Stuttgart

Selected Research Projects and Activities *(1/2)*

- **Dynamic analysis of ocean models**
  - Architecture Recovery from Fortran Code

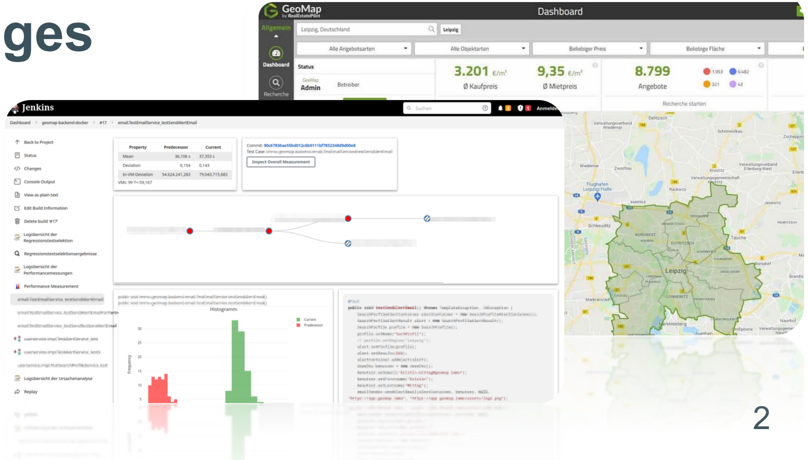
Presentation by
Reiner today



© Geomar

- **PeASS: Identifying Performance Changes**
  - Geomap industrial case study

Presentation by
David tomorrow



Property	Predecessor	Current
Mean	36,106 s	37,353 s
Deviation	0,154	0,143
In-VM-Deviation	54.624.241,283	79.043.715,683

VMs: 99 T=-59,167

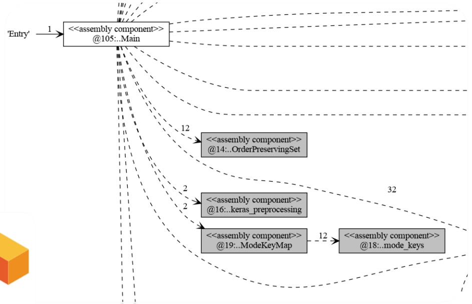
Commit: 90c67836ae55bd012c6b9111bf7852348d9d00e8

Test Case:

© Geomap

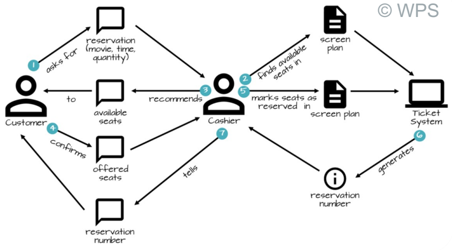
Selected Research Projects and Activities *(2/2)*

- **Performance evaluation of data-intensive systems**
  - Instrumentation of AI-enabled systems with Python/Keras/TensorFlow


'Entry'
1
<<assembly component>> @105:..Main
12
<<assembly component>> @14:..OrderPreservingSet
2
2
32
<<assembly component>> @16:..keras_preprocessing
<<assembly component>> @19:..ModeKeyMap
12
<<assembly component>> @18:..mode_keys

Presentation by Serafim today

- **Domain-centric runtime quality analysis**
  - Domain-centric monitoring (starting from DDD)



© WPS

Presentation by Heiko tomorrow

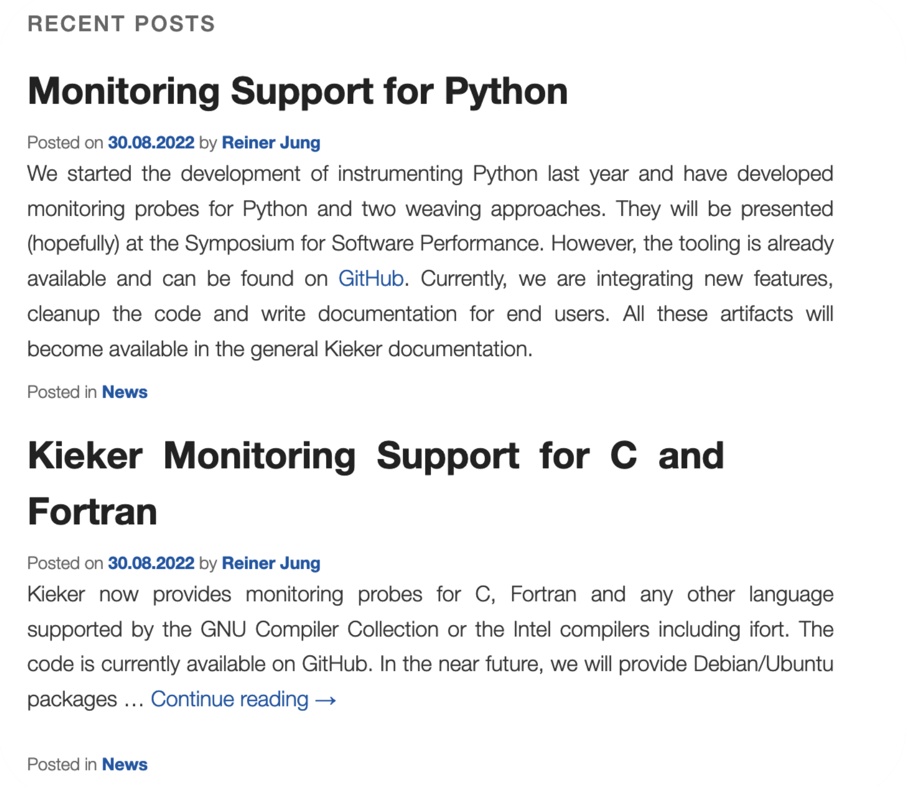
Extended Language Support for Non-Java Languages

Python

Fortran

C / C++

Presentations by Serafim + Reiner today



RECENT POSTS

Monitoring Support for Python

Posted on 30.08.2022 by Reiner Jung

We started the development of instrumenting Python last year and have developed monitoring probes for Python and two weaving approaches. They will be presented (hopefully) at the Symposium for Software Performance. However, the tooling is already available and can be found on GitHub. Currently, we are integrating new features, cleanup the code and write documentation for end users. All these artifacts will become available in the general Kieker documentation.

Posted in News

Kieker Monitoring Support for C and Fortran

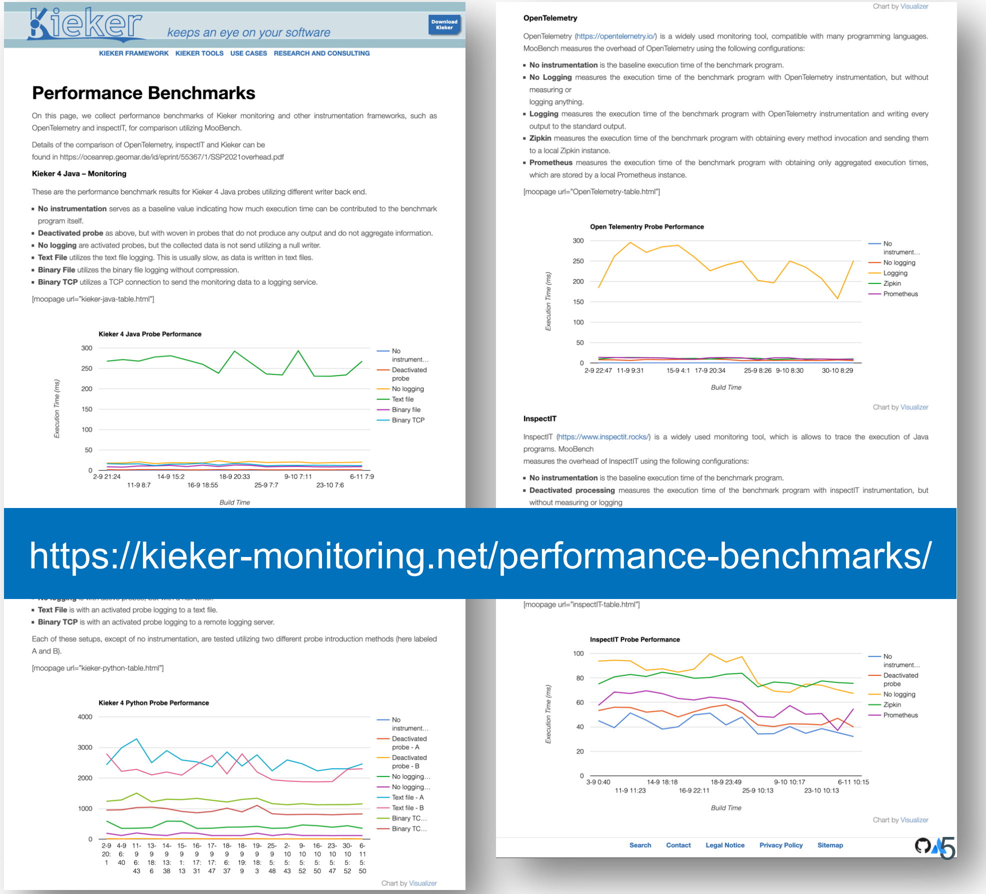
Posted on 30.08.2022 by Reiner Jung

Kieker now provides monitoring probes for C, Fortran and any other language supported by the GNU Compiler Collection or the Intel compilers including ifort. The code is currently available on GitHub. In the near future, we will provide Debian/Ubuntu packages … Continue reading →

Posted in News

Updates to MooBench Benchmark

- Updated to
  - current Kieker version
  - OpenTelemetry
  - inspectIT
- Continuous execution of benchmarks has been automated (again)
- Now includes Kieker4Python
- Planned for C/Fortran/C++



Performance Benchmarks

On this page, we collect performance benchmarks of Kieker monitoring and other instrumentation frameworks, such as OpenTelemetry and inspectIT, for comparison utilizing MooBench.

Details of the comparison of OpenTelemetry, inspectIT and Kieker can be
found in https://oceanrep.geomar.de/id/eprint/55367/1/SSP2021overhead.pdf

Kieker 4 Java – Monitoring

These are the performance benchmark results for Kieker 4 Java probes utilizing different writer back end.

- **No instrumentation** serves as a baseline value indicating how much execution time can be contributed to the benchmark program itself.
- **Deactivated probe** as above, but with woven in probes that do not produce any output and do not aggregate information.
- **No logging** are activated probes, but the collected data is not send utilizing a null writer.
- **Text File** utilizes the text file logging. This is usually slow, as data is written in text files.
- **Binary File** utilizes the binary file logging without compression.
- **Binary TCP** utilizes a TCP connection to send the monitoring data to a logging service.

[moopage url="kieker-java-table.html"]

Chart by Visualizer

OpenTelemetry

OpenTelemetry (https://opentelemetry.io/) is a widely used monitoring tool, compatible with many programming languages. MooBench measures the overhead of OpenTelemetry using the following configurations:

- **No instrumentation** is the baseline execution time of the benchmark program.
- **No Logging** measures the execution time of the benchmark program with OpenTelemetry instrumentation, but without measuring or
logging anything.
- **Logging** measures the execution time of the benchmark program with OpenTelemetry instrumentation and writing every output to the standard output.
- **Zipkin** measures the execution time of the benchmark program with obtaining every method invocation and sending them to a local Zipkin instance.
- **Prometheus** measures the execution time of the benchmark program with obtaining only aggregated execution times, which are stored by a local Prometheus instance.

[moopage url="OpenTelemetry-table.html"]

Chart by Visualizer

InspectIT

InspectIT (https://www.inspectit.rocks/) is a widely used monitoring tool, which is allows to trace the execution of Java programs. MooBench
measures the overhead of InspectIT using the following configurations:

- **No instrumentation** is the baseline execution time of the benchmark program.
- **Deactivated processing** measures the execution time of the benchmark program with inspectIT instrumentation, but without measuring or logging

https://kieker-monitoring.net/performance-benchmarks/

- **Text File** is with an activated probe logging to a text file.
- **Binary TCP** is with an activated probe logging to a remote logging server.

Each of these setups, except of no instrumentation, are tested utilizing two different probe introduction methods (here labeled A and B).

[moopage url="kieker-python-table.html"]

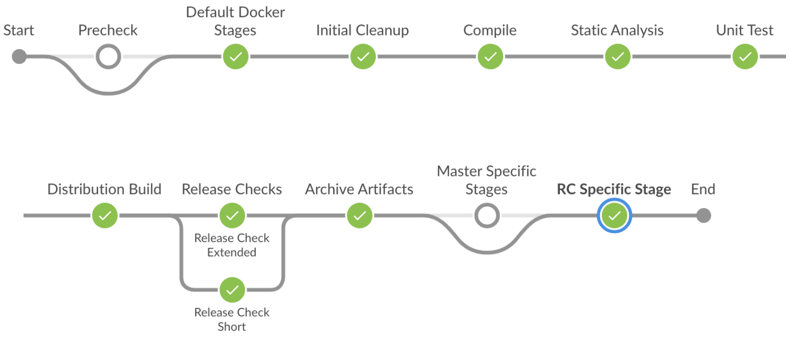
Chart by Visualizer

[moopage url="inspectIT-table.html"]

Chart by Visualizer

Releases and Release Process

- Changed (continuous) release policy
  - Semantic versioning: ***`MAJOR.MINOR.FIX`***
  - Extended automation towards continuous delivery (via Maven central)
  - Latest 1.15.2 on Nov 7, 2022



- Upcoming: Kieker 2.0
  - Finalized integration of TeeTime-based stages
  - Consistent naming conventions for stages
  - Restructured packages: technology-based → topic-based
  - Revised architecture model and architecture analysis
  - Observe and analyze user behavior based on graph clustering algorithms

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CAU
Christian-Albrechts-Universität zu Kiel

HITeC

Lancaster University
Leipzig

Kieker

https://kieker-monitoring.net/

Update @ SSP 2022