Utilizing Performance Unit Tests
To Increase Performance Awareness

Vojtěch Horký, Peter Libič, Lukáš Marek,
Antonín Steinhauser and Petr Tůma

February 4, 2015

Charles University in Prague
Motivation: Choosing a Plotting Library

We need to plot graphs for our web application in Java.

Our choice is driven by various requirements.

- Available features
- Price
- Sane and documented API
- Performance
- ...
Motivation: Choosing a Plotting Library

We need to plot graphs for our web application in Java.

Our choice is driven by various requirements.

- Available features
- Price
- Sane and documented API
- Performance
- ...

Department of Distributed and Dependable Systems
Motivation: Choosing a Plotting Library Based on its Performance

Comparison when plotting 800x600 PNG image

- GRAL
- XChart
- JFreeChart

Execution time [ms] vs. Number of data points plotted
Issues With Low-Impact Performance Decisions

To decide which of the libraries is faster we had to

– design and implement a test
– and evaluate the results.
Issues With Low-Impact Performance Decisions

To decide which of the libraries is faster we had to

- design and implement a test
- and evaluate the results.

This takes time. Often, we

- assume the performance differences are negligible
- or fallback to previous experience with similar task.
Our Goal

Help the developer with decisions that have low performance impact.

– Without extra effort from the developer.
– Give the answers as fast as possible.

Make developers aware of the actual performance of their code.

(We do not aim to correct bad architectural & design decisions.)
The Idea: Extend API Documentation with Performance Information

Why API documentation?

– We target methods and classes.
– Available even in IDE as context help.
The Idea: Extend API Documentation with Performance Information

Why API documentation?
- We target methods and classes.
- Available even in IDE as context help.

How it would be used?

When coding, developers would see the performance information together with the method detail.
indexOf

public int indexOf(Object o)

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. More formally, returns the lowest index i such that (0 <= i < length) and object.equals(elementAt(i)) is true. Returns -1 if this list contains no element equivalent to the specified element.

Specified by:
indexOf in interface List<E>

Overrides:
indexOf in class AbstractList<E>

Parameters:
- o - element to search for

Returns:
- index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element

lastIndexOf

public int lastIndexOf(Object o)

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element. More formally, returns the highest index i such that (0 <= i < length) and object.equals(elementAt(i)) is true. Returns -1 if this list contains no element equivalent to the specified element.

Specified by:
lastIndexOf in interface List<E>

Overrides:
lastIndexOf in class AbstractList<E>

Parameters:
- o - element to search for

Returns:
- index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element

clone

public Object clone()

Returns a shallow copy of this ArrayList instance. (The elements themselves are not copied.)

Overrides:
clone in class Object

Returns:
Tools: ... to PerfJavaDoc

contains

public boolean contains(Object o)

Returns true if this list contains the specified element. More formally, returns true if and only if this list contains at least one element e such that (o == null ? e == null : o.equals(e)).

Specified by:
- contains in interface Collection&lt;E&gt;

Specified by:
- contains in interface List&lt;E&gt;

Overrides:
- contains in class AbstractCollection&lt;E&gt;

Parameters:
- e - element whose presence in this list is to be tested

Returns:
- true if this list contains the specified element

Performance:

Workload: Unsuccessful search

Tests performance of a search in the collection.

The workload generator is usable for any class implementing the Collection interface.

Collection size
600 — 3000

Search time [μs] vs Collection size

Apply
Getting Workloads from Performance Unit Tests

- Performance Unit Test
  - Workload Parameters
    - Method Selection
      - Measurements
        - Workload
          - Method Under Test
            - Test Harness
  - Test Condition
    - Condition Evaluation
  - Method Under Test
    - Test Harness
Getting Workloads from Performance Unit Tests

- Performance Unit Test
- Workload Parameters
- Workload Generator
- Test Harness
- Method Under Test
- Measurements
- Method Selection
- Condition Evaluation
- Test Condition
- Condition Evaluation
Getting Workloads from Performance Unit Tests
Getting Workloads from Performance Unit Tests

- Performance Unit Test
- Workload Parameters
- Workload Generator
- Test Harness
- Workload
- Method Under Test

- Condition Evaluation
- Method Selection
- Measurements
- Test Condition

Department of Distributed and Dependable Systems
Getting Workloads from Performance Unit Tests

1. Performance Unit Test
2. Workload Parameters
3. Workload Generator
4. Test Condition
5. Method Selection
6. Measurements
7. Condition Evaluation
8. Workload
9. Method Under Test
10. Test Harness
Getting Workloads from Performance Unit Tests
Getting Workloads from Performance Unit Tests

- Performance Unit Test
- Workload Parameters
- Workload Generator
- Method Under Test
- Test Harness
- Test Condition
- Method Selection
- Measurements
- Condition Evaluation
- Workload
Getting Workloads from Performance Unit Tests
Getting Workloads from Performance Unit Tests

- Performance Unit Test
- Workload Parameters
- Workload Generator
- Test Condition
- Method Selection
- Workload
- Method Under Test
- Test Harness
- Condition Evaluation
- Measurements
Getting Workloads from Performance Unit Tests

This provides data for the performance documentation.
From Performance Unit Tests to Documentation

Connecting the workload to the method

```c
void sort(long[] data) {
    ...
}
```
From Performance Unit Tests to Documentation

Connecting the workload to the method

```java
@Workload("pkg.Workload.longArray")
void sort(long[] data) {
    ...
}
```
From Performance Unit Tests to Documentation

Connecting the workload to the method

```java
@Workload("pkg.Workload.longArray")
void sort(long[] data) {
    ...
}
```

```java
long[] longArray(
    int size)
    ...
}
```
Connecting the workload to the method

```java
@Workload("pkg.Workload.longArray")
void sort(long[] data) {
    ...
}
```

Getting labels for the plots

```java
long[] longArray(int size) {
    ...
}
```
Connecting the workload to the method

```java
@Workload("pkg.Workload.longArray")
void sort(long[] data) {
    ...
}
```

Getting labels for the plots

```java
@Descr("Generate array filled with random longs")
long[] longArray(int size) {
    ...
}
```
Connecting the workload to the method

```java
@Workload("pkg.Workload.longArray")
void sort(long[] data) {
    ...
}
```

Getting labels for the plots

```java
@Descr("Generate array filled with random longs")
long[] longArray(@Param("Array size") int size) {
    ...
}
```
Displaying Performance Interactively

– Measure on demand
– Cache & share the results
– Refine results continuously (on background)
Displaying Performance Interactively

– Measure on demand
– Cache & share the results
– Refine results continuously (on background)
Experiment: Faster Applications with Extended Documentation

Volunteer students (advanced course of Java).

- Groups with and without performance documentation.
- Task to solve with Java JDOM library.
Experiment: Faster Applications with Extended Documentation

Volunteer students (advanced course of Java).

- Groups with and without performance documentation.
- Task to solve with Java JDOM library.

Results

Participants have problems judging performance of their own code: expected run-times differed in orders of magnitude.

Not feasible to decide whether one group writes faster applications (we would need thousands of students to have statistically reliable comparison of the groups).
Experiment: Improve Existing Applications

Simulate developers caring about performance.

Use similar – but faster – methods guided by results from the extended documentation.
Experiment: Improve Existing Applications

Simulate developers caring about performance.

Use similar – but faster – methods guided by results from the extended documentation.

Applications

– Buildhealth
  – Reports merged results from various test frameworks.

– METS Downloader
  – Gathers bibliographic meta-information.

(Selected because of a reasonable size and use of JDOM.)
## Improving Existing Applications: Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildhealth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>METS downloader</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Improving Existing Applications: Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildhealth</td>
<td>Changed code</td>
<td></td>
</tr>
<tr>
<td>METS downloader</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Improving Existing Applications: Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildhealth</td>
<td>Changed code</td>
<td>938.3 ms</td>
</tr>
<tr>
<td>METS downloader</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Improving Existing Applications: Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildhealth</td>
<td>938.3 ms</td>
<td>908.4 ms</td>
</tr>
<tr>
<td>Whole application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>METS downloader</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Improving Existing Applications: Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildhealth</td>
<td>938.3 ms</td>
<td>908.4 ms</td>
</tr>
<tr>
<td>Changed code</td>
<td>2.52 s</td>
<td>2.40 s</td>
</tr>
<tr>
<td>Whole application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>METS downloader</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Improving Existing Applications: Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Buildhealth</strong></td>
<td>Changed code</td>
<td>938.3 ms</td>
</tr>
<tr>
<td>Whole application</td>
<td>2.52 s</td>
<td>2.40 s</td>
</tr>
<tr>
<td><strong>METS downloader</strong></td>
<td>Changed code</td>
<td></td>
</tr>
</tbody>
</table>
# Improving Existing Applications: Results

## Application | Before | After
--- | --- | ---
Buildhealth | 938.3 ms | 908.4 ms
  | 2.52 s | 2.40 s
METS downloader | 131.5 ms | 27.4 ms
## Improving Existing Applications: Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Buildhealth</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changed code</td>
<td>938.3 ms</td>
<td>908.4 ms</td>
</tr>
<tr>
<td>Whole application</td>
<td>2.52 s</td>
<td>2.40 s</td>
</tr>
<tr>
<td><strong>METS downloader</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changed code</td>
<td>131.5 ms</td>
<td>27.4 ms</td>
</tr>
<tr>
<td>Whole application</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Improving Existing Applications: Results

<table>
<thead>
<tr>
<th>Application</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Buildhealth</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changed code</td>
<td>938.3 ms</td>
<td>908.4 ms</td>
</tr>
<tr>
<td>Whole application</td>
<td>2.52 s</td>
<td>2.40 s</td>
</tr>
<tr>
<td><strong>METS downloader</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changed code</td>
<td>131.5 ms</td>
<td>27.4 ms</td>
</tr>
<tr>
<td>Whole application</td>
<td>120.2 s</td>
<td>120.4 s</td>
</tr>
</tbody>
</table>
Conclusion: Utilizing Performance Unit Tests To Increase Performance Awareness

Extend the API documentation and make the developers aware of performance of small parts of the code and help them write faster applications.

http://d3s.mff.cuni.cz/software/spl
Conclusion: Utilizing Performance Unit Tests To Increase Performance Awareness

Extend the API documentation and make the developers aware of performance of small parts of the code and help them write faster applications.

http://d3s.mff.cuni.cz/software/spl

Thank You!