WORKFLOW ENGINE PERFORMANCE BENCHMARKING WITH BENCHFLOW
The BenchFlow Project

“Design and implement the first benchmark to assess and compare the performance of WfMSs that are compliant with Business Process Model and Notation 2.0 standard.”
What is a Workflow Management System?
Many Vendors of BPMN 2.0 WfMSs

https://en.wikipedia.org/wiki/List_of_BPMN_2.0_engines

<table>
<thead>
<tr>
<th>Year</th>
<th>Number of BPMN 2.0 WfMSs</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2010</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2011</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>2012</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>2013</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>2014</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>2015</td>
<td>2</td>
<td>23</td>
</tr>
<tr>
<td>2016</td>
<td>0</td>
<td>23</td>
</tr>
</tbody>
</table>

Grand Total: 23

BETA

BPMN 2.0  
Aug 2009

BPMN 2.0  
Jan 2011

ISO/IEC 19510

BPMN 2.0.2  
Jan 2014
Why do We Need a Benchmark?

*end-users, vendors, developers*
Why do We Need a Benchmark?  
*end-users, vendors, developers*

1. How to choose the best WfMS according to the company’s technical requirements?

2. How to choose the best WfMS according to the company’s business process models (workflows)?
Why do We Need a Benchmark?  
*end-users, vendors, developers*

1. How to choose the best WfMS according to the company’s technical requirements?

2. How to choose the best WfMS according to the company’s business process models (workflows)?

3. How to evaluate performance improvements during WfMS’s development?

4. How to identify WfMS’s bottlenecks?
The BenchFlow Benchmarking Process

*Input/Process/Output Model*

- **Workload Model**
  - Workload
  - WfMS
  - Measure
- **Configurations**
- **Metrics KPIs**
The BenchFlow Benchmarking Process

*Input/Process/Output Model*

**Workload Model**

**Workload Mix**

- **A**: 80%
- **B**: 20%
- **C**: 80%
- **D**: 20%

**Configurations**

**Metrics KPIs**
The BenchFlow Benchmarking Process

*Input/Process/Output Model*

- **Workload Mix**
  - 80% of Workload Mix
  - 20% of Workload Mix

- **Workload Model**
  - Test Types
  - Test Data
  - Load Functions

- **Configurations**
  - Workload
  - WfMS
  - Measure

- **Metrics KPIs**
The BenchFlow Benchmarking Process

*Input/Process/Output Model*

**Workload Model**

**Workload Mix**

- **A**: 80%
- **B**: 20%
- **C**: 80%
- **D**: 20%

**Test Types**

- Input/Process/Output Model

**WfMS**

**Configurations**

**Test Data**

- Load Functions
- Containers

**Measure**

**Metrics KPIs**
The BenchFlow Benchmarking Process

**Input/Process/Output Model**

- **Workload Model**
  - **Workload Mix**
    - 80% C, A, B
    - 20% D
  - **Test Data**
  - **Load Functions**
  - **Test Types**

- **WfMS Configurations**

- **Performance Data**
  - **Metrics KPIs**
    - Throughput
    - Process Instance Duration

- **Measure**
  - **Containers**
Container Based Methodology and Framework

Methodology

Framework
Container Based Methodology and Framework

Methodology

Provides Great Advantages for Automation and Reproducibility of Results, while Ensuring Negligible Performance Impact
Container-centric Methodology for Benchmarking Workflow Management Systems.

[CLOSER '16]

[Image: Diagram showing the benchmarking methodology in a workflow.]

- **Agreement Proposal:** BenchFlow agrees on adding Vendor's WfMS to the Benchmark.
- **Containerized WfMS Request:** BenchFlow requests Containerized Distribution of WfMS.
- **Verified Benchmark Results:** BenchFlow verifies the results. If valid, Publish Benchmark Results. If not, Results Verification Outcome.

Vendor, Community, and BenchFlow are involved in the process.
BenchFlow Benchmarking Methodology

requirements from the WfMS

Availability of APIs

• Deploy Process
• Start Process Instance
• Users API
• Web Service APIs
• Events APIs
BenchFlow Benchmarking Methodology

requirements from the WfMS

Availability of APIs

• Deploy Process
• Start Process Instance
• Users API
• Web Service APIs
• Events APIs

Availability of Timing Data

• Workflow & Construct:
  • Start Time
  • End Time
  • [Duration]
**BenchFlow Benchmarking Methodology**

*requirements from the WfMS*

**Availability of APIs**
- Deploy Process
- Start Process Instance
- Users API
- Web Service APIs
- Events APIs

**Availability of Timing Data**
- Workflow & Construct:
  - Start Time
  - End Time
  - [Duration]

**Availability of Execution State**
State of the workflow execution. E.g., running, completed, error
BenchFlow Framework

WfMSs’ specific characteristics

• Manages the Automatic WfMS deployment;
• Provides a “plugin” mechanism to add new WfMSs;
• Performs automatic Process Models deployment;
• Collects Client and Server-side data and metrics;
• Automatically computes performance metrics and statistics.
Applications of the Methodology and Framework

Micro-Benchmarking BPMN 2.0 Workflow Management Systems with Workflow Patterns.

[CAiSE ’16]
Micro-Benchmarking with Workflow Patterns

research questions

1. What is the impact of individual or a mix of workflow patterns on the performance of each one of the benchmarked BPMN 2.0 WfMSs?

2. Are there performance bottlenecks for the selected WfMSs?

Selected three WfMSs: popular open-source WfMSs actually used in industry
Micro-Benchmarking with Workflow Patterns

workload mix

Sequence Pattern [SEQ]

Pattern Reference:
Micro-Benchmarking with Workflow Patterns

workload mix

Sequence Pattern [SEQ]

Exclusive Choice and Simple Merge [EXC]

Pattern Reference:
Micro-Benchmarking with Workflow Patterns

workload mix

Arbitrary Cycle [CYC]

Pattern Reference:

Micro-Benchmarking with Workflow Patterns

workload mix

Parallel Split and Synchronisation [PAR]

Pattern Reference:
Micro-Benchmarking with Workflow Patterns

workload mix

Parallel Split and Synchronisation [PAR]

Explicit Termination Pattern [EXT]

Pattern Reference:
Micro-Benchmarking with Workflow Patterns

workload mix design decisions

1. Maximise the simplicity of the model expressing the workflow pattern

2. Omit the interactions with external systems by implementing all tasks as script tasks
Micro-Benchmarking with Workflow Patterns

Load Functions

Test Type

Load test

<table>
<thead>
<tr>
<th># of Instance Producers</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>Think Time</td>
<td>1 sec</td>
</tr>
<tr>
<td>Max User req/sec (r)</td>
<td>1</td>
</tr>
<tr>
<td>Load Time (T_l)</td>
<td>10 min</td>
</tr>
<tr>
<td>Ramp-Up Period (T_r)</td>
<td>30 sec</td>
</tr>
<tr>
<td>Connection Time-out (T_o)</td>
<td>20 sec</td>
</tr>
</tbody>
</table>

Example for u=1500
Micro-Benchmarking with Workflow Patterns

configurations: WfMS environments

WfMS A

WfMS B

WfMS C
Micro-Benchmarking with Workflow Patterns

configurations: WfMS environments

MySQL: Community Server 5.6.26
Micro-Benchmarking with Workflow Patterns

configurations: WfMS environments

MySQL: Community Server 5.6.26

WfMS A
O.S.: Ubuntu 14.04.01
J.V.M.: Oracle Serv. 7u79

WfMS B
Ubuntu 14.04.01
Oracle Serv. 7u79

WfMS C
Ubuntu 14.04.01
Oracle Serv. 7u79
Micro-Benchmarking with Workflow Patterns

configurations: WfMS environments

**MySQL**: Community Server 5.6.26

**Os.:** Ubuntu 14.04.01

**J.V.M.:** Oracle Serv. 7u79

**App. Server:** Ap. Tomcat 7.0.62

**WfMS A**

MySQL

**WfMS B**

MySQL

**WfMS C**

MySQL

Ubuntu 14.04.01

Oracle Serv. 7u79

Ap. Tomcat 7.0.62

Ubuntu 14.04.01

Oracle Serv. 7u79

Wildfly 8.1.0. Final
Micro-Benchmarking with Workflow Patterns

configurations: WfMS environments

MySQL: Community Server 5.6.26

O.S.: Ubuntu 14.04.01
J.V.M.: Oracle Serv. 7u79

Max Java Heap: 32GB
Micro-Benchmarking with Workflow Patterns

configurations: WfMS environments

MySQL: Community Server 5.6.26

WfMS A

MySQL

WfMS B

MySQL

WfMS C

MySQL

O.S.: Ubuntu 14.04.01
J.V.M.: Oracle Serv. 7u79
Max Java Heap: 32GB

Max DB Connections Number: 100
Micro-Benchmarking with Workflow Patterns

configurations: WfMS environments

MySQL: Community Server 5.6.26

WfMS A

MySQL

O.S.: Ubuntu 14.04.01
J.V.M.: Oracle Serv. 7u79
Max Java Heap: 32GB

WfMS B

MySQL

O.S.: Ubuntu 14.04.01
J.V.M.: Oracle Serv. 7u79
Max DB Connections Number: 100

WfMS C

MySQL

O.S.: Ubuntu 14.04.01
J.V.M.: Oracle Serv. 7u79
App. Server: Wildfly 8.1.0. Final

WfMS’s Containers: official ones, if available on DockerHub
Micro-Benchmarks with Workflow Patterns

configurations: WfMS environments

**MySQL**: Community Server 5.6.26

---

**WfMS A**

- **O.S.**: Ubuntu 14.04.01
- **J.V.M.**: Oracle Serv. 7u79
- **App. Server**: Ap. Tomcat 7.0.62
- **Max Java Heap**: 32GB

**WfMS B**

- **O.S.**: Ubuntu 14.04.01
- **J.V.M.**: Oracle Serv. 7u79
- **App. Server**: Ap. Tomcat 7.0.62
- **Max Java Heap**: 32GB

**WfMS C**

- **O.S.**: Ubuntu 14.04.01
- **J.V.M.**: Oracle Serv. 7u79
- **App. Server**: Wildfly 8.1.0. Final
- **Max Java Heap**: 32GB

**Max DB Connections Number**: 100

**WfMS’s Containers**: official ones, if available on DockerHub

**WfMS’s Configuration**: as suggested by vendor's documentation
## Micro-Benchmarking with Workflow Patterns

*configurations: WfMS deployment*

<table>
<thead>
<tr>
<th></th>
<th>Load Drivers</th>
<th>WfMS</th>
<th>DBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>64 Cores</td>
<td>12 Cores</td>
<td>64 Cores</td>
</tr>
<tr>
<td></td>
<td>@ 1400 MHz</td>
<td>@ 800 MHz</td>
<td>@ 2300 MHz</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>128 GB</td>
<td>64 GB</td>
<td>128 GB</td>
</tr>
</tbody>
</table>

**TEST ENVIRONMENT**
## Micro-Benchmarking with Workflow Patterns

*configurations: WfMS deployment*

### Test Environment

<table>
<thead>
<tr>
<th>Component</th>
<th>CPU</th>
<th>RAM</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Drivers</td>
<td>64 Cores @ 1400 MHz</td>
<td>128 GB</td>
<td>10 Gbit/s</td>
</tr>
<tr>
<td>WfMS</td>
<td>12 Cores @ 800 MHz</td>
<td>64 GB</td>
<td>10 Gbit/s</td>
</tr>
<tr>
<td>DBMS</td>
<td>64 Cores @ 2300 MHz</td>
<td>128 GB</td>
<td>10 Gbit/s</td>
</tr>
</tbody>
</table>
## Micro-Benchmarking with Workflow Patterns

**configurations: WfMS deployment**

### Load Drivers
- **CPU**: 64 Cores @ 1400 MHz
- **RAM**: 128 GB

### WfMS
- **CPU**: 12 Cores @ 800 MHz
- **RAM**: 64 GB

### DBMS
- **CPU**: 64 Cores @ 2300 MHz
- **RAM**: 128 GB
- **Network**: 10 Gbit/s

### Test Environment

- **O.S.**: Ubuntu 14.04.3 LTS

---
Micro-Benchmarking with Workflow Patterns

configurations: WfMS deployment

**Load Drivers**
- **CPU**: 64 Cores @ 1400 MHz
- **RAM**: 128 GB

**WfMS**
- **CPU**: 12 Cores @ 800 MHz
- **RAM**: 64 GB

**DBMS**
- **CPU**: 64 Cores @ 2300 MHz
- **RAM**: 128 GB

**TEST ENVIRONMENT**

- **CPU**
  - 10 Gbit/s
- **RAM**
  - 10 Gbit/s

**O.S.:**
- Ubuntu 14.04.3 LTS
- 1.8.2
Micro-Benchmarking with Workflow Patterns

performed experiments

Individual patterns

1. 1500 (max) concurrent Instance Producers starting each pattern
   [SEQ, EXC, CYC, EXT, PAR]

Mix of patterns

2. 1500 concurrent Instance Producers starting an equal number of
   pattern instances (20% mix of [SEQ, EXC, CYC, EXT, PAR])
Micro-Benchmarking with Workflow Patterns

performed experiments

Individual patterns

1. 1500 (max) concurrent Instance Producers starting each pattern
   \([\text{SEQ, EXC, CYC, EXT, PAR}]\)

Mix of patterns

2. 1500 concurrent Instance Producers starting an equal number of
   pattern instances (20% mix of \([\text{SEQ, EXC, CYC, EXT, PAR}]\))

Three runs for each execution of the experiments
Micro-Benchmarking with Workflow Patterns

performed experiments

Individual patterns

1. 1500 (max) concurrent Instance Producers starting each pattern [SEQ, EXC, CYC, EXT, PAR]

Mix of patterns

2. 1500 concurrent Instance Producers starting an equal number of pattern instances (20% mix of [SEQ, EXC, CYC, EXT, PAR])

Three runs for each execution of the experiments

The data collected in the first minute of the execution has not been included in the analysis
Micro-Benchmarking with Workflow Patterns
performed experiments

Individual patterns

1. 1500 (max) concurrent Instance Producers starting each pattern
   [SEQ, EXC, CYC, EXT, PAR]

Mix of patterns

2. 1500 concurrent Instance Producers starting an equal number of
   pattern instances (20% mix of [SEQ, EXC, CYC, EXT, PAR])

Three runs for each execution of the experiments
The data collected in the first minute of the execution has not been included in the analysis
Computed Metrics and Statistics

*engine level metrics*

- Throughput
- Database Size
- Execution Time
- Number of Process Instances

Diagram:
- Load Driver
- WfMS
- Application Server
- Database
- DBMS
- Users
- Web Service
Computed Metrics and Statistics

engine level metrics

- Throughput
- Database Size
- Execution Time
- Number of Process Instances
...
### Micro-Benchmarking with Workflow Patterns

*individual patterns: throughput (bp/sec)*

<table>
<thead>
<tr>
<th></th>
<th>SEQ</th>
<th>EXC</th>
<th>EXT</th>
<th>PAR</th>
<th>CYC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WfMS A</td>
<td>1456.79</td>
<td>1417.17</td>
<td>1455.68</td>
<td>1433.12</td>
<td>327.99</td>
</tr>
<tr>
<td>WfMS B</td>
<td>1447.66</td>
<td>1436.03</td>
<td>1426.35</td>
<td>1429.65</td>
<td>644.02</td>
</tr>
<tr>
<td>WfMS C</td>
<td>63.31</td>
<td>49.04</td>
<td>0.38</td>
<td>48.89</td>
<td>13.46</td>
</tr>
</tbody>
</table>

- **WfMS A** at $u=600$ throughput is 327.99 bp/sec.
- **WfMS B** at $u=800$ throughput is 644.02 bp/sec.
- **WfMS C** at $u=1500$ throughput is 13.46 bp/sec.
Micro-Benchmarking with Workflow Patterns

*individual patterns: WfMS C Scalability for SEQ pattern*

WfMS C uses a synchronous instantiation API, load drivers must wait for workflows to end before starting new ones
Micro-Benchmarking with Workflow Patterns

*individual patterns: WfMS C Scalability for SEQ pattern*

**WfMS C uses a synchronous instantiation API, load drivers must wait for workflows to end before starting new ones**

![Graph showing response time and throughput against instance producers](image-url)
Micro-Benchmarking with Workflow Patterns

*individual patterns: WfMS C Scalability for SEQ pattern*

**WfMS C** uses a synchronous instantiation API, load drivers must wait for workflows to end before starting new ones.

We have found a bottleneck! (~60 bp/sec)
Computed Metrics and Statistics

*process level metrics*

- Throughput
- Instance Duration
- ...
Computed Metrics and Statistics

*process level metrics*

- **Throughput**
- **Instance Duration**

---

**Load Driver**

**WfMS**

**Application Server**

**DBMS**

**Users**

**Web Service**

**Per Process Definition**

**Throughput**
Micro-Benchmarking with Workflow Patterns

*individual patterns: mean (instance duration time)*
Micro-Benchmarking with Workflow Patterns

*individual patterns: mean (instance duration time)*

![Diagram showing sequence patterns and their durations for WfMS A, B, and C.](image)
Micro-Benchmarking with Workflow Patterns

*individual patterns: mean (instance duration time)*

![Diagram showing comparisons between Workflow Management Systems (WfMS) A, B, and C for different patterns: SEQ, EXC, EXT, PAR, CYC.](image)

Duration (ms):
- SEQ: 0.74, 0.39, 0.85, 0.48, 0.4
- EXC: 6.39, 9.3, 0.4
- EXT: 14.1, 0.7
- PAR: 13.29, 10.06, 3.06
- CYC: 2.91

[EXT]
Micro-Benchmarking with Workflow Patterns

individual patterns: mean (instance duration time)
Micro-Benchmarking with Workflow Patterns

*individual patterns: mean (instance duration time)*

CYC Instance Producers:

\[ u = 600 \]
Computed Metrics and Statistics

environment metrics

Load Driver

CPU/RAM Usage per Second

IO Operations/Network Usage per Second

Web Service

Users

DBMS

Instance Database

Application Server

WfMS
Computed Metrics and Statistics

environment metrics

CPU/RAM Usage per Second

IO Operations/Network Usage per Second
Micro-Benchmarking with Workflow Patterns

individual patterns: mean (RAM) and mean (CPU) usage
Micro-Benchmarking with Workflow Patterns

*performed experiments*

Individual patterns

1. 1500 (max) concurrent Instance Producers starting each pattern 
   \([SEQ, EXC, CYC, EXT, PAR]\)

Mix of patterns

2. 1500 concurrent Instance Producers starting an equal number of pattern instances (20% mix of \([SEQ, EXC, CYC, EXT, PAR]\))

Three runs for each execution of the experiments

The data collected in the first minute of the execution has not been included in the analysis
Micro-Benchmarking with Workflow Patterns

mix of patterns: duration, RAM, CPU

- Duration: MIX 540.02 ms, WfMS A 1.22 ms, WfMS B 8.16 ms
- RAM: MIX 2,786.54 MB, WfMS A 991.86 MB, WfMS B 2,310.88 MB
- CPU: MIX 48.53%, WfMS A 48.53%, WfMS B 77.69%
Micro-Benchmarking with Workflow Patterns

*summary*

Even though the Executed Workflows are Simple
Micro-Benchmarking with Workflow Patterns

*summary*

Even though the Executed Workflows are Simple

There are relevant differences in workflows’ duration and throughput among the WfMSs
Micro-Benchmarking with Workflow Patterns

**summary**

Even though the Executed Workflows are Simple

There are relevant differences in workflows’ duration and throughput among the WfMSs

There are relevant differences in resource usage among WfMSs
Micro-Benchmarking with Workflow Patterns

**summary**

Even though the Executed Workflows are Simple

There are relevant differences in workflows’ duration and throughput among the WfMSs

There are relevant differences in resource usage among WfMSs

One WfMS has bottlenecks
Highlights
Highlights

Benchmarking Process

Benchmarking Methodology
Highlights

BenchFlow Benchmarking Process

BenchFlow Framework

BenchFlow Benchmarking Methodology
### Highlights

**BenchFlow Benchmarking Process**

![Diagram of the BenchFlow Benchmarking Process](image1.png)

**BenchFlow Framework**

![Diagram of the BenchFlow Framework](image2.png)

**Benchmarking Methodology**

![Diagram of the Benchmarking Methodology](image3.png)

**Micro-Benchmark with WP**

![Diagram of the Micro-Benchmark with Workflow Patterns](image4.png)

The BenchFlow Benchmarking Process

- **Input/Process/Output Model**
- **Workload Model**
  - **Workload Mix**
  - **Test Types**
  - **Test Data Load Functions**
- **Configurations**
- **WMS**
- **Performance Data**
- **Measurement**
  - **Metrics KPIs**
  - **Throughput**
  - **Process Instance Duration**

**BenchFlow Framework**

- **Architecture**
  - **Servers**
  - **Containers**
- **WMS**
- **Analyzers**
  - **Performance Metrics**
  - **Performance KPIs**
- **Data Transformers**
  - **Spark**
  - **Hadoop**
  - **Docker**
- **COLLECTORS**
  - **MONITOR**
  - **Adapters**
- **Adapters**
  - **Faban Drivers**
- **Analysis**
  - **DBMS**
  - **Instance Database**
- **HARNESS**
  - **Web Service**
  - **Minio**

**Micro-Benchmark with Workflow Patterns**

- **Individual patterns: mean (instance duration time)**
- **CYC Instance Producers:**
  - **u=600**
- **Data**
  - **Web Service**
  - **Minio**

---

**Benchmarking Process**

- **Load Functions**
- **Test Data**
- **Performance**
- **Workload Mix**
- **KPIs**
- **Input/Process/Output Model**
- **Containers**
- **BenchFlow Framework architecture**
- **BenchFlow Benchmarking Process**
- **Micro-Benchmarking with Workflow Patterns**
- **Individual patterns: mean (instance duration time)**
- **Data**
- **Web Service**
- **Minio**

---

**BenchFlow Framework**

- **Architecture**
  - **Servers**
  - **Containers**
- **WMS**
- **Analyzers**
  - **Performance Metrics**
  - **Performance KPIs**
- **Data Transformers**
  - **Spark**
  - **Hadoop**
  - **Docker**
- **COLLECTORS**
  - **MONITOR**
  - **Adapters**
- **Adapters**
  - **Faban Drivers**
- **Analysis**
  - **DBMS**
  - **Instance Database**
- **HARNESS**
  - **Web Service**
  - **Minio**

---

**Micro-Benchmark with WP**

- **Individual patterns: mean (instance duration time)**
- **CYC Instance Producers:**
  - **u=600**
- **Data**
  - **Web Service**
  - **Minio**
Call for Collaboration

WfMSs, process models, process logs

**WfMSs**

- We want to add more and more WfMSs to the benchmark
- **Contact us for collaboration, and BenchFlow framework support**

**Process Models**

- We want to characterise the Workload Mix using Real-World process models
- **Share your executable BPMN 2.0 process models, even anonymised**

**Execution Logs**

- We want to characterise the Load Functions using Real-World behaviours
- **Share your execution logs, even anonymised**
WORKFLOW ENGINE
PERFORMANCE BENCHMARKING
WITH BENCHFLOW

benchflow
http://benchflow.inf.usi.ch

Vincenzo Ferme
Faculty of Informatics
USI Lugano, Switzerland
BACKUP SLIDES

Vincenzo Ferme
Faculty of Informatics
USI Lugano, Switzerland
Published Work

[SSP ’14]

[BTW ’15]

[ICPE ’15]
Published Work

[CLOSER ’15]

[SOSE ’15]

[BPM ’15]
Published Work

[BPMD ’15]

[ICPE ’16]

[CLOSER ’16]
Published Work

[ICWE '16]

[CAiSE '16]

[ICWS '16]
Published Work

[SummerSOC ’16]

[BPM Forum ’16]

[OTM ’16]
Docker Performance

[IBM ’14]

“Our results show that containers result in equal or better performance than VMs in almost all cases.”

“Although containers themselves have almost no overhead, Docker is not without performance gotchas. Docker volumes have noticeably better performance than files stored in AUFS. Docker’s NAT also introduces overhead for workloads with high packet rates. These features represent a tradeoff between ease of management and performance and should be considered on a case-by-case basis.”

BenchFlow Configures Docker for Performance by Default
Benchmarking Requirements

- Relevant
- Representative
- Portable
- Scalable
- Simple
- Repeatable
- Vendor-neutral
- Accessible
- Efficient
- Affordable

- S. E. Sim, S. Easterbrook et al., *Using benchmarking to advance research: A challenge to software engineering*, 2003
BenchFlow Benchmarking Methodology

requirements from the WfMS

CORE

NON-CORE
BenchFlow Benchmarking Methodology

requirements from the WfMS

Initialisation APIs

- Deploy Process
- Start Process
- Instance

WfMS

CORE

NON-CORE
BenchFlow Benchmarking Methodology

requirements from the WfMS

Initialisation APIs

Deploy Process
Start Process Instance

User APIs

Create User
Pending User Tasks
Claim Task
Complete Task

CORE
NON-CORE
BenchFlow Benchmarking Methodology

requirements from the WfMS

Initialisation APIs
- Deploy Process
- Start Process

User APIs
- Create User
- Pending User Tasks
- Claim Task
- Complete Task

Web Service APIs
- Create Group
- Invoke WS

WfMS

CORE

NON-CORE
BenchFlow Benchmarking Methodology

requirements from the WfMS

Initialisation APIs

User APIs

Web Service APIs

Event APIs

Deploy Process

Create User

Pending User Tasks

Claim Task

Complete Task

Pending Event Tasks

Create Group

invoke WS

Complete Task

Issue Event

WfMS

WfMS

WfMS

CORE

NON-CORE

Start Process Instance
### BenchFlow Benchmarking Methodology

**requirements from the WfMS**

<table>
<thead>
<tr>
<th>Core APIs</th>
<th>Functionality</th>
<th>Min Response Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initialisation APIs</strong></td>
<td>Deploy a process</td>
<td>Deployed process ID</td>
</tr>
<tr>
<td></td>
<td>Start a process instance</td>
<td>Process instance ID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-core APIs</th>
<th>Functionality</th>
<th>Min Response Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User APIs</strong></td>
<td>Create a user</td>
<td>User ID</td>
</tr>
<tr>
<td></td>
<td>Create a group of users</td>
<td>User group ID</td>
</tr>
<tr>
<td></td>
<td>Access pending tasks</td>
<td>Pending tasks IDs</td>
</tr>
<tr>
<td></td>
<td>Claim a task*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complete a task</td>
<td></td>
</tr>
<tr>
<td><strong>Event APIs</strong></td>
<td>Access pending events</td>
<td>Pending events IDs</td>
</tr>
<tr>
<td></td>
<td>Issue events</td>
<td></td>
</tr>
<tr>
<td><strong>Web service APIs</strong></td>
<td>Map tasks to Web service endpoints</td>
<td></td>
</tr>
</tbody>
</table>

*Optional depending on the WfMS implementation*
BenchFlow Framework

*system under test*

Docker Engine
BenchFlow Framework

system under test

Docker Engine

Containers
BenchFlow Framework

*system under test*

**Docker Machine**

provides

**Docker Engine**

Containers
BenchFlow Framework

system under test

Docker Machine

provides

Docker Swarm

Docker Engine

Containers
BenchFlow Framework

system under test

Docker Machine

provides

manages

Containers

Docker Swarm

Servers

Docker Engine
BenchFlow Framework

system under test

Docker Machine

provides

Docker Compose

manages

Docker Swarm

Containers

Docker Engine

Servers
BenchFlow Framework

system under test

Docker Machine

Docker Compose
SUT’s Deployment Conf.

Docker Engine
Containers

Docker Swarm
Servers

provides

manages

deploys
Server-side Data and Metrics Collection

asynchronous execution of workflows
Server-side Data and Metrics Collection

asynchronous execution of workflows
Server-side Data and Metrics Collection

asynchronous execution of workflows
Server-side Data and Metrics Collection

monitors
Server-side Data and Metrics Collection

monitors
Server-side Data and Metrics Collection

*monitors*

Monitors’ Characteristics:
- RESTful services
- Lightweight (written in Go)
- As less invasive on the SUT as possible

Examples of Monitors:
- CPU usage
- Database state
Server-side Data and Metrics Collection

**monitors**

---

**Monitors’ Characteristics:**
- RESTful services
- Lightweight (written in Go)
- As less invasive on the SUT as possible

**Examples of Monitors:**
- CPU usage
- Database state
Server-side Data and Metrics Collection

collect data

Test Execution

harness

Web Service

WfMS

DBMS

Servers

Containers

Faban Drivers
Server-side Data and Metrics Collection

collect data

- Servers
- Containers
- Web Service
- WfMS
- DBMS
- Faban Drivers
- harness
- Minio
- Instance Database
Server-side Data and Metrics Collection

collect data

Collectors’ Characteristics:
- RESTful services
- Lightweight (written in Go)
- Two types: online and offline
- Buffer data locally

Examples of Collectors:
- Container's Stats (e.g., CPU usage)
- Database dump
- Applications Logs
Server-side Data and Metrics Collection

**collect data**

---

Collectors’ Characteristics:
- RESTful services
- Lightweight (written in Go)
- Two types: online and offline
- Buffer data locally

Examples of Collectors:
- Container's Stats (e.g., CPU usage)
- Database dump
- Applications Logs
Synchronisation with the Analyses

schedule analyses

A high-throughput distributed messaging system
Synchronisation with the Analyses

Schedule analyses

A high-throughput distributed messaging system

Test Execution

Analyses
Synchronisation with the Analyses

schedule analyses

A high-throughput distributed messaging system

Test Execution

Analyses
Synchronisation with the Analyses

schedule analyses

A high-throughput distributed messaging system

Test Execution

Servers Containers

harness

Faban Drivers

Web Service

Stats Collector

DBMS

DB Collector

WfMS

Analyses

Subscribe

A high-throughput distributed messaging system

kafka

Spark

COLLECT

Publish

Instance Database

Minio
Synchronisation with the Analyses

schedule analyses

A high-throughput distributed messaging system

Test Execution

Servers

Containers

Web Service

WfMS

Stats Collector

DB Collector

DBMS

Analyses

Subscribe

kafka

Spark

Read

Minio

Instance Database

Faban Drivers

Containers

Collect

Publish

Read
Performance Metrics and KPIs

amount of data

Number of Tests

Number of Repetitions

Amount of Data (GB)
Performance Metrics and KPIs

amount of data

Number of Tests

Number of Repetitions

Amount of Data (GB)

1

Number of WfMSs

1 3 5 10 20 50 100

1 3 5 10

52
# Micro-Benchmarking with Workflow Patterns

*mix of patterns: throughput (bp/sec)*

<table>
<thead>
<tr>
<th>WfMS</th>
<th>MIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>WfMS A</td>
<td>1061.27</td>
</tr>
<tr>
<td>WfMS B</td>
<td>1402.33</td>
</tr>
<tr>
<td>WfMS C</td>
<td>1.78</td>
</tr>
</tbody>
</table>
Computed Metrics and Statistics

feature level metrics

Load Driver

Application Server

Instance Duration

Throughput

DBMS

Instance Duration

Per Construct (type, name)

WfMS

Users

Web Service

DBMS
Computed Metrics and Statistics

feature level metrics

- Load Driver
- Users
- Web Service
- Application Server
- DBMS

Per Construct (type, name):
- Instance Duration
- Throughput

…
Computed Metrics and Statistics

*interactions metrics*

Response Time

Latency
Computed Metrics and Statistics

*interactions metrics*

**Response Time**

**Latency**

...
Computed Metrics and Statistics

descriptive and homogeneity statistics

- Descriptive Statistics (e.g., Mean, Confidence Interval, Standard Deviation, Percentiles, Quartiles, …)
- Coefficient of Variation across trials
- Levene Test on trials