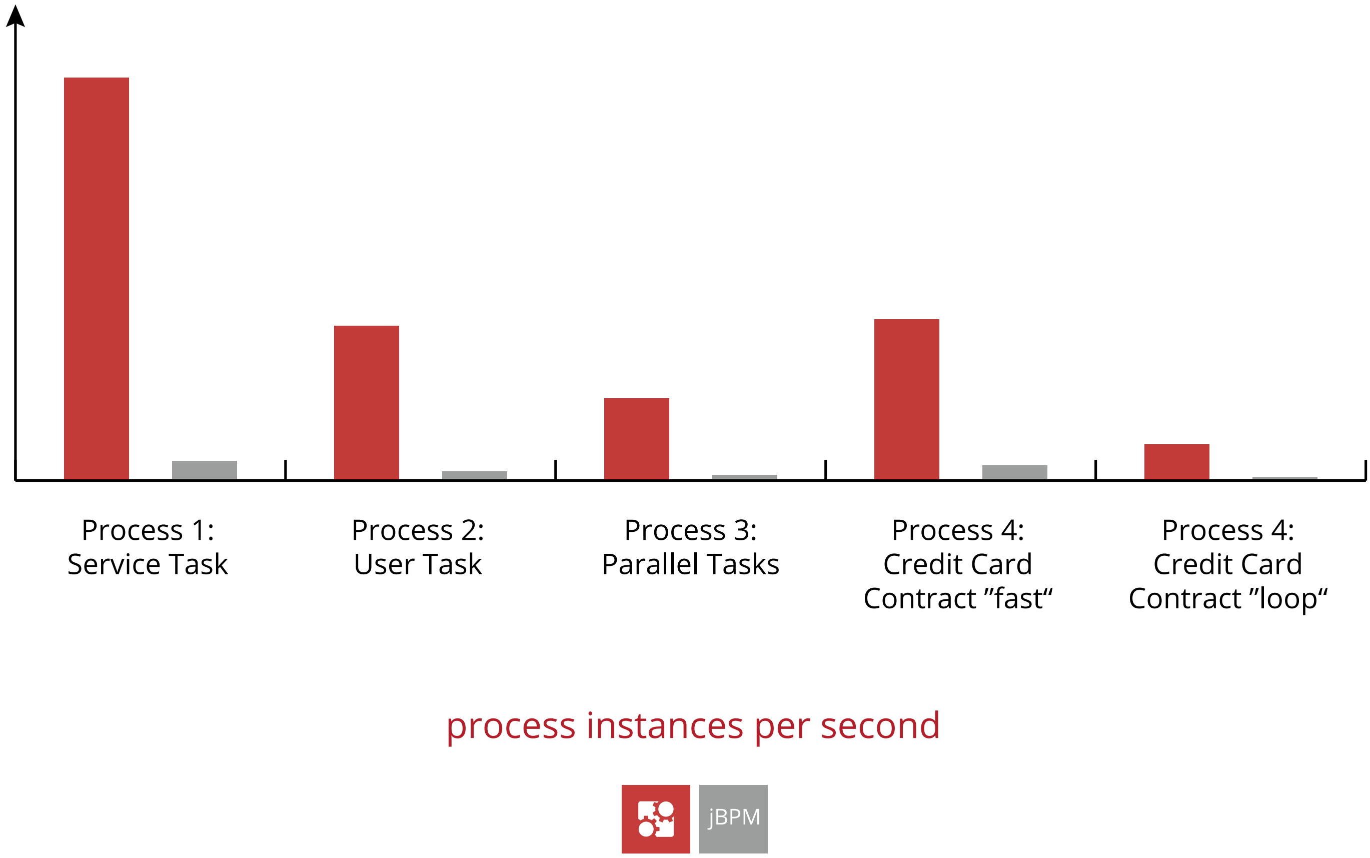
camunda
the business process company

Camunda BPM 7 compared to JBoss jBPM 6



October 2014 – camunda.com

Contents

Introduction

This whitepaper compares Camunda BPM 7 to JBoss jBPM 6.

The significant findings are:

- Camunda BPM 7 strategically aims for “Developer-Friendliness”, whilst JBoss jBPM 6 strives for the “Zero-Code-BPM”-ideal.
- Camunda BPM 7 offers innovative, powerful features that are missing from JBoss jBPM 6.
- According to our tests, Camunda BPM 7 performs 10x better than JBoss jBPM 6.
- The support and professional services around Camunda BPM are carried out by highly experienced experts.
- The pricing model for the Camunda BPM Enterprise Edition is flexible and allows a low cost entry.

The information used for this comparison is mostly publicly available and referenced accordingly (documentation, etc.).

The performance comparison is based on extensive testing. The concrete test structure will be clearly described, the source code of the test framework used will be linked.

Company

Camunda, based out of Berlin, Germany, was founded in March 2008 and currently employs around 30 full time employees. The corporation is managed by the founders and owners and has not raised any external funding (venture capital, loans, etc.). It has been profitable since the founding date and has grown organically on an average of 50% per year. This has led to remarkably stable leadership which, in both short and long term perspectives, benefits our customers.

Camunda has been focusing on the topic of business process management (BPM) since day one and participated in defining the BPMN 2.0 standard. The book “Real-Life BPMN”, which was published in 2009, has become one of the most successful publications on BPMN and is available in English, Spanish and German. After four years of focused consulting on BPM, in 2012 we decided to offer our own BPM software product. The extraordinary success thereof has led to a majority of our revenue being generated by Camunda BPM and us defining ourselves primarily as a software vendor. In spring 2014 Camunda Inc. was founded, based in San Francisco, California. Since then, it has successfully been developing the North American market with quality software, “Made in Germany”.

The Red Hat, Inc. develops JBoss jBPM. Red Hat sells over 50 software products, the core product being a commercial distribution of the Linux operating system. As a publicly traded company, Red Hat must also look out for shareholder interests.

Products

Product visions

“Technology Overview”[1], published by Red Hat, says about “Red Hat JBoss BPM-Suite 6”:

»One of the main design objectives for Red Hat JBoss BPM Suite has always been to empower non-technical users to define and automate business processes without compromising flexibility or scalability. This vision has led Red Hat to a fully model-driven approach to managing business processes, business data, and forms, and for creating advanced dashboards for business activity monitoring (BAM).«

The subsequent sections describe how a fully model-driven application development based on the product can occur.

To enable this, JBoss jBPM obviously follows the paradigm of a proprietary BPM Suite, which provides a vendor-specific form of software development. This is also known as “Zero-Code BPM”. Even the available source code cannot reduce the associated negative consequences for software development. These are explained in detail in Sandy Kemsley‘s whitepaper “The myth of Zero-Code BPM”. Kemsley concludes:

»When creating complex, core business process applications, “zero code” doesn’t mean “zero developers”, and a proprietary BPMS development environment can hinder enterprise developers. Lightweight developer-friendly BPM that integrates with existing enterprise environments can be a better fit with corporate development standards, requires less training for developers and has less vendor lock-in.«

Said “lightweight, developer-friendly BPM” is exactly Camunda‘s product vision and determines our strategic product design. Our customers agree, as assessed by Hamburger Sparkasse, the leading bank for medium-sized businesses in Northern Germany:

»Camunda BPM offers a promising open-source implementation of BPMN 2.0. In addition to open-source support, its lightweight solution and developer friendliness were further important factors for a decision in favor of Camunda BPM.«

The second key aspect of our product vision is the business-IT alignment based on BPMN. This is not achieved by enabling the business departments to program without any IT professionals – which is in itself contradictory. But by an intelligent combination of directly executable, easy to read BPMN diagrams with classic software development. S-Kreditpartner GmbH states:

»Our expectations that experts from the business side and IT can work together based on the BPMN 2.0. standards were fully met. Camunda is the key player in the development and establishment of the BPMN 2.0 standards. Camunda BPM offers us a complete software stack that fully covers our needs.«

[1] http://www.redhat.com/en/files/resources/en-rhjb-bpm-suite-6-tech-overview-11347767.pdf

Unfortunately those in favor of developer friendliness are frequently underestimated during the initial evaluation of BPM products. Especially the business departments are often tempted to use a model-driven BPM-Suite, which seemingly allows greater independence from the IT. This approach is almost always doomed to fail, as numerous companies can confirm. It is therefore no coincidence that more than 50% of Camunda Enterprise customers have replaced a previously used “Zero Code”-BPM Suite.

Features

A detailed feature comparison of both products cannot be carried out in this whitepaper for two reasons: For one, it is beyond the scope of this document. Secondly, both products experience continuous development. A feature that may be missing today, could already be included tomorrow.

Our recommendation is therefore an internal proof of concept for both products and a test of relevant features for your project. In this case, you should make sure to take into account aspects of the development process, e.g. how intuitive and well documented are the provided APIs.

In this document we can only provide a snapshot and point out some basic features that Camunda BPM 7 has and that are missing from JBoss jBPM 6 in its current version (as of October 2014):

CMMN: Besides BPMN 2.0, Camunda BPM also supports CMMN 1.0. This standard has been defined by the OMG, which is also responsible for BPMN and UML. CMMN enables the so-called “Case Management” practice which supports the processing of unstructured procedures. This means that tasks can be dynamically added, updated or skipped within a sequence. Thanks to Camunda BPM‘s unique combination of BPMN and CMMN, you can develop applications that seamlessly combine structured workflows with Case Management.

Cockpit: Camunda Cockpit is an extensive HTML 5 web application for process operation. Thanks to the modular architecture of the cockpit you can develop individual plugins that can be integrated as widgets into the user interface. Users also benefit from the great open source community that continuously develops new plugins and makes them publicly available.

Container-Support: Camunda BPM supported application servers include Tomcat, JBoss AS / EAP, IBM Websphere Application Server and Oracle Weblogic Server. Support for the Oracle Weblogic Server is currently not available in JBoss jBPM 6.[2]

[2] https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_BPM_Suite/6.0/html/Installation_Guide/Supported_platforms.html

Performance

For this whitepaper the behavior of jBPM 6.0.0.Final under load was compared with the behavior of Camunda BPM 7.1.0. It thus allows an assessment of the performance of the two products in direct comparison.

Architecture of the performance tests

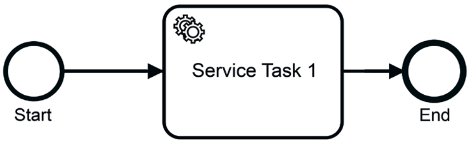
The tests were conducted in the following runtime environment:

- **Hardware:** Thinkpad T420s with Intel Core i5-2520M @ 2.50 GHz, 8 GB memory, hard drive SAMSUNG SSD 830 Series
- **Operating system:** Windows 7 Professional
- **Virtual Machine:** Oracle OTN Developer Day VM with Oracle Linux 6.5
- **Database:** Oracle Database 12c Release 1 Enterprise Edition in virtual machine.

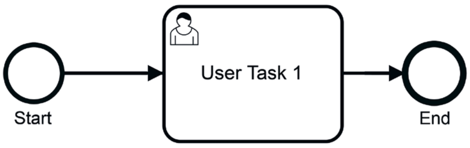
Tested processes

The following processes were tested:

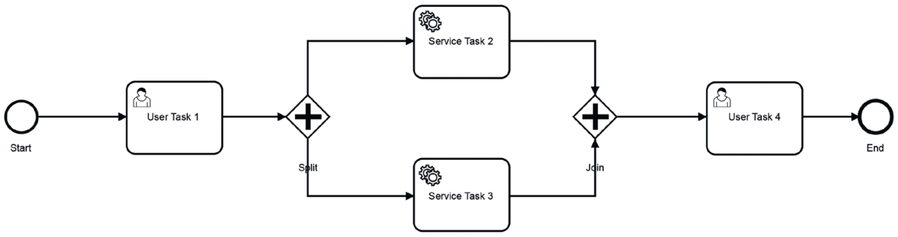
Process 1 – Service Task: A process that consists solely of one service task.



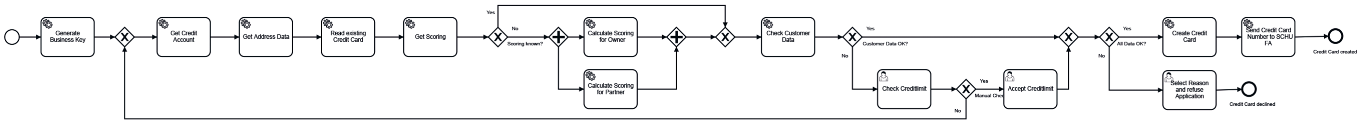
Process 2 – User Task: A process that consists solely of one user task.



Process 3 – Parallel Tasks: A process that intends for a parallel execution of service tasks in between two user tasks.



Process 4 – Credit Card: A real customer process that provides a complex execution of different tasks. It can be run in different variations. For our tests, we used the variations "Fast" (direct run) and "Loop" (run incl. loop).



They were created in the BPMN standard in jBPM-eclipse-plugin. For execution in Camunda the following camunda-specific attributes were added:

- Attribute camunda:class="…" for the implementation of service tasks.
- Element <camunda:taskListener class="…" event="create"> to save the task-ID in the thread context. This means that the user task can be completed without prior search for the task in the database.

For decisions on conditional flows after gateways platform-dependent forms were used. In Camunda the decision reads as:

```
<bpmn2:conditionExpression xsi:type="bpmn2:tFormalExpression" id="FormalExpression_43">
       #{scoringKnown == true}
</bpmn2:conditionExpression>
```

In jBPM the condition needs to be defined as follows:

```
<bpmn2:conditionExpression xsi:type="bpmn2:tFormalExpression" id="FormalExpression_43">
       return scoringKnown == true;
</bpmn2:conditionExpression>
```

All processes were executed out with a performance-testing framework.[3] To start jBPM from the framework, a few classes were refactored.

The implementation of the service tasks stays empty for both engines, as the performance of the engine is to be measured without external influences.

[3] https://github.com/camunda/camunda-bpm-platform/tree/master/qa/performance-tests-engine

Testing method

For each process flow, a test was developed, which passes through the entire process. For each measurement, the test was performed at least five times and the measured figures then formed the average ratio.

Process 3, for example, appears as follows:

```
@Test @Deployment(resources="UserAndServiceTaskPerformanceTest.
userParallelServiceUserTask.bpmn")
        public void userParallelServiceUserTask() {
                performanceTest()
                .step(new StartProcessInstanceStep(engine, "UserAndServiceTaskProcess"))
                .step(new CompleteTaskStep(engine, PerfTestConstants.TASK_ID))
                .step(new CompleteTaskStep(engine, PerfTestConstants.TASK_ID))
                .run();
        }
```

This test is carried out by the framework 100 or 1,000 times. The first run uses one thread to simulate a user. During the second run, two threads execute 100 processes parallel in order to simulate two users and to find out how well the engine scales.

The path leading through process 4 was controlled by process variables. This means that loops can be created if the first test of customer data shows inaccuracies. Once the data has been corrected, it will be accepted by a user during the second test. The following example shows the test for jBPM:

```
@Test public void createCreditCardLoop() {
        Map<String, Object> creditCardData = new HashMap<String, Object>();
        creditCardData.put("scoringKnown", Boolean.FALSE);
        creditCardData.put("customerDataOK", Boolean.FALSE);
        creditCardData.put("manualCheckOK", Boolean.TRUE);
        creditCardData.put("allDataOK", Boolean.TRUE);
        Map<String, Object> manualCheckOK = new HashMap<String, Object>();
        manualCheckOK.put("manualCheckResult", Boolean.TRUE);
        Map<String, Object> manualCheckNotOK = new HashMap<String, Object>();
        manualCheckNotOK.put("manualCheckResult", Boolean.FALSE);
        performanceTest() .step(new StartProcessInstanceStep(getRuntimeManager(),
                "CreditCardContract",
                creditCardData)) .step(new CompleteTaskStep(getRuntimeManager(),
                PerfTestConstants.TASK_ID,
                manualCheckNotOK)) .step(new CompleteTaskStep(getRuntimeManager(),
                PerfTestConstants.TASK_ID,
                manualCheckOK)) .step(new CompleteTaskStep(getRuntimeManager(),
                PerfTestConstants.TASK_ID))
                .run();
        }
```

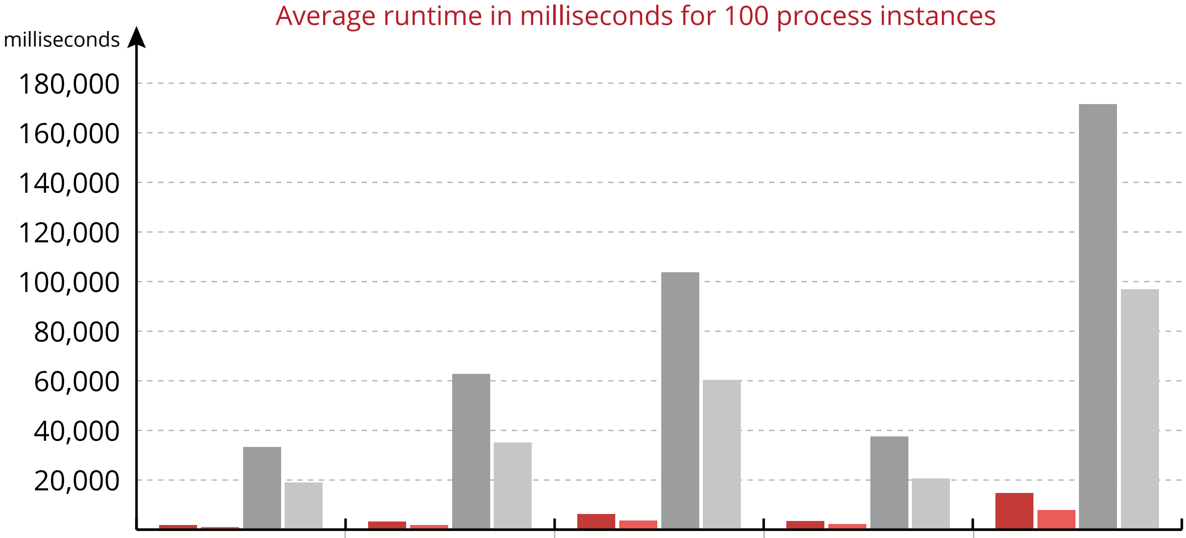
Performance – 100 process instances

The performance metrics below show the results of 100 process executions by Camunda BPM and jBPM.

It should be noted that the runtime is reduced (or the throughput increased) if not only one, but two threads are started for execution.

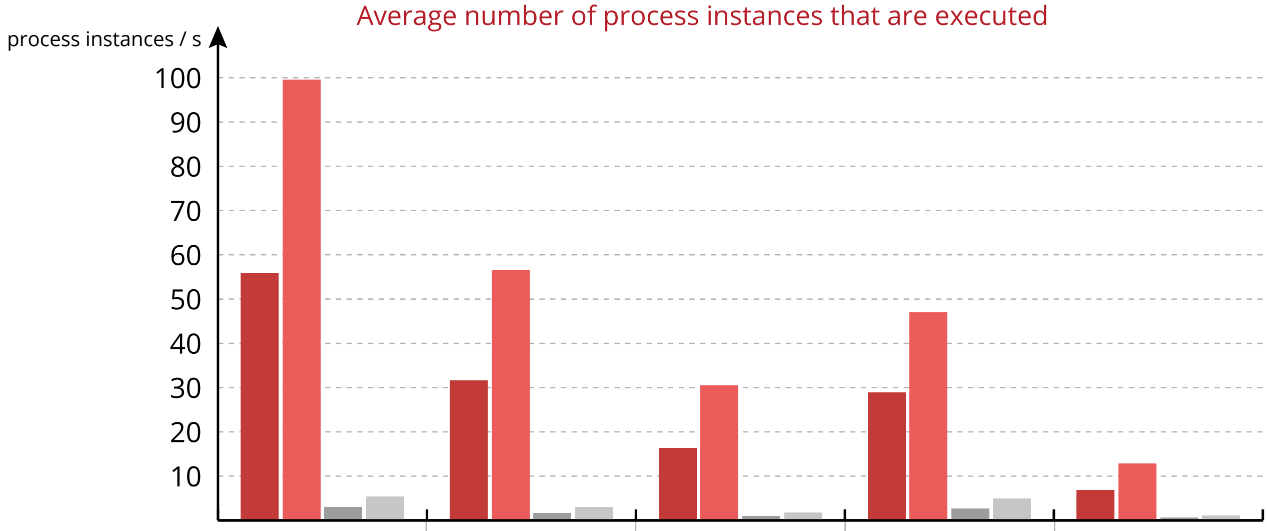
In all situations, Camunda BPM is significantly more efficient than jBPM, i.e. the time that is needed for execution of the 100 process instances is significantly shorter (depending on the test by a factor of 10-20). Conversely, significantly more process instances can be exectued per second. “Credit Card Contract fast” means that the main path was used during the process – theoretically, due to the contained XOR gateways, more paths are possible. For comparison, the “Loop” has been run through once.

Average runtime in milliseconds for 100 process instances



	Process 1: Service Task	Process 2: User Task	Process 3: Parallel Tasks	Process 4: Credit Card Contract “fast”	Process 4: Credit Card Contract “loop”
🟥 Camunda 1 Thread	1,820.00	3,209.40	6,237.20	3,481.00	14,751.80
🟥 Camunda 2 Threads	1,029.80	1,919.20	3,582.00	2,162.60	7,921.60
■ jBPM 1 Thread	33,228.60	62,743.60	103,664.80	37,487.00	171,601.00
□ jBPM 2 Threads	19,021.40	35,146.00	60,253.60	20,582.80	96,827.40

Average number of process instances that are executed

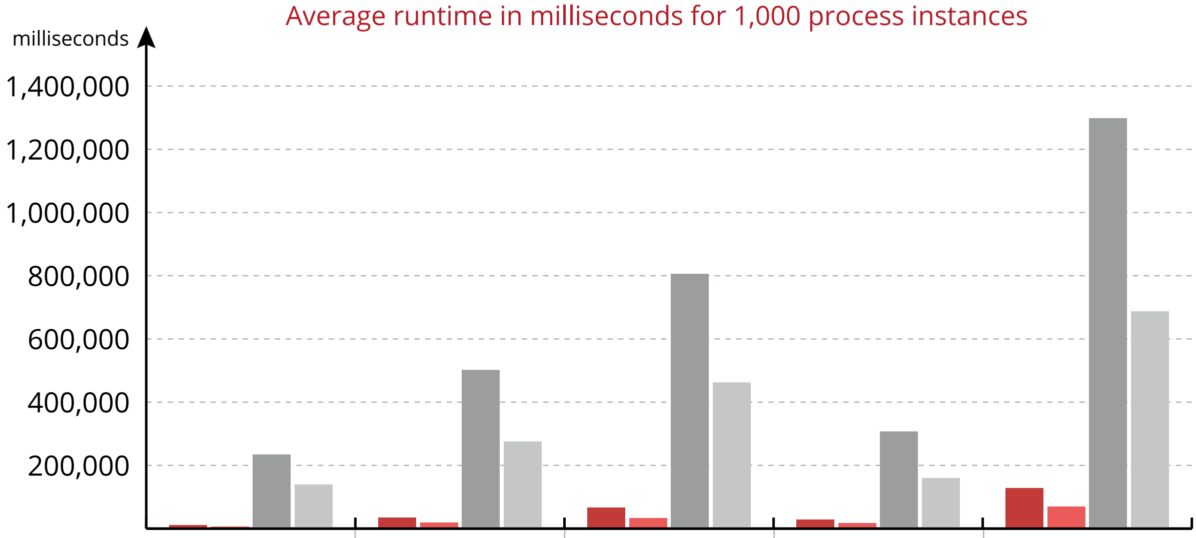


	Process 1: Service Task	Process 2: User Task	Process 3: Parallel Tasks	Process 4: Credit Card Contract “fast”	Process 4: Credit Card Contract “loop”
🟥 **Camunda 1 Thread**	55.86	31.64	16.33	28.90	6.84
🟥 **Camunda 2 Threads**	99.60	56.58	30.47	47.03	12.83
⬛ **jBPM 1 Thread**	3.04	1.64	1.00	2.69	0.60
⬜ **jBPM 2 Threads**	5.32	2.96	1.71	4.91	1.05

Performance – 1,000 process instances

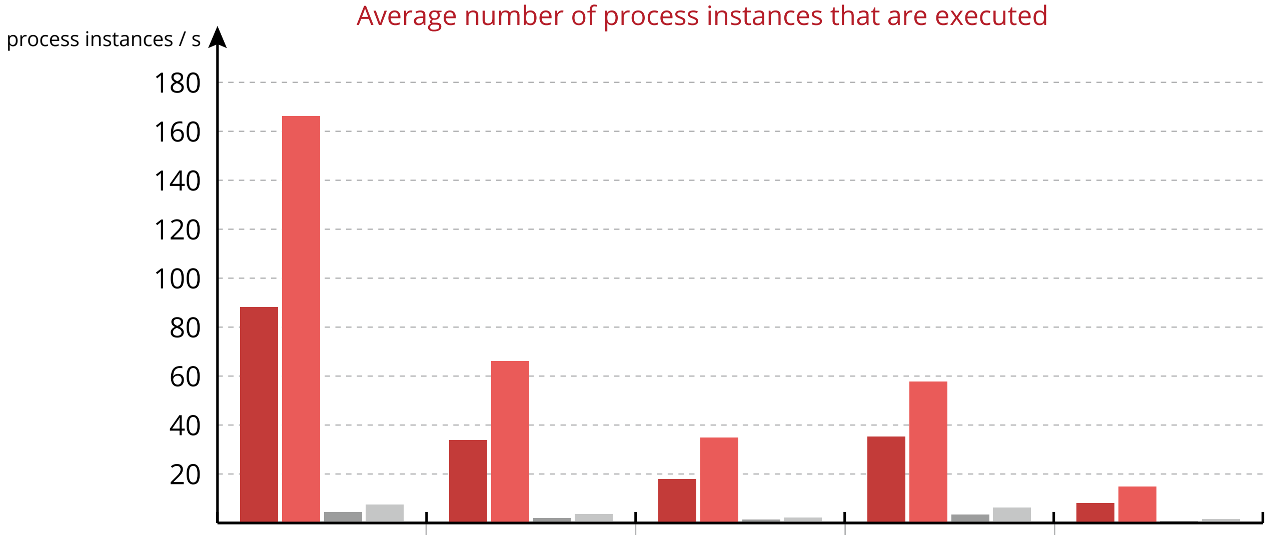
If the number of running process instances is increased tenfold, the performance advantage of Camunda remains significant.

Average runtime in milliseconds for 1,000 process instances



	Process 1: Service Task	Process 2: User Task	Process 3: Parallel Tasks	Process 4: Credit Card Contract “fast”	Process 4: Credit Card Contract “loop”
■ Camunda 1 Thread	11,403.70	35,136.30	66,283.00	28,429.10	128,791.20
■ Camunda 2 Threads	6,038.30	18,478.00	32,996.80	17,405.20	69,799.30
■ jBPM 1 Thread	234,137.89	501,403.44	806,431.44	307,568.33	1,298,534.56
■ jBPM 2 Threads	139,012.44	276,115.78	463,039.44	160,562.56	687,032.22

Average number of process instances that are executed



	Process 1: Service Task	Process 2: User Task	Process 3: Parallel Tasks	Process 4: Credit Card Contract “fast”	Process 4: Credit Card Contract “loop”
🟥 Camunda 1 Thread	88.04	33.85	17.94	35.25	7.97
🟥 Camunda 2 Threads	166.27	66.09	34.82	57.65	14.87
⬛ jBPM 1 Thread	4.29	2.01	1.26	3.30	0.78
⬜ jBPM 2 Threads	7.50	3.66	2.19	6.29	1.48

Conclusion

The behavior of both engines is in principle comparable. If a request is made via the interface, this request is processed directly in a thread until the result is returned. If a process is started, for example, which can run directly (first measurement, only services), the method does not end until all steps of the process have been completed and stored in the history. If the process is to branch off into a user task, the method is terminated as soon as the user task is stored in the database. It can then be found in a new external access. This behavior is described in the following manuals: jBPM User Guide 6.0.1, chapter 5 Core Engine API, section 5.3.4 Threads[4] and Camunda BPM Userguide, chapter Process Engine, section Transaction in Processes.[5]

From these descriptions and measurements we conclude that Camunda BPM 7 is in terms of performance much more efficiently implemented than JBoss jBPM 6.

[4] http://docs.jboss.org/jbpm/v6.0.1/userguide/jBPMCoreEngine.html#d0e1883

[5] http://docs.camunda.org/latest/guides/user-guide/#process-engine-transactions-in-processes

Services

Support

The successful implementation of a BPM platform depends largely on the vendor support. This does not imply onsite consulting services but the continuous support for any questions or problems that is guaranteed with a service level agreement. This is usually done by phone and / or ticket system.

Support for Camunda BPM comes directly from the company‘s headquarter in Berlin. No call centers are involved. All employees in the support team are fluent in English and sit in the same rooms as the product‘s core developers who can be involved in the query whenever necessary.

During the evaluation of a BPM product, you should therefore always check the support. Direct contact to the provider‘s reference customers can be very helpful as well.

Camunda regularly receives excellent feedback for its support. The references below can be contacted directly at any time if required:

iTradeNetwork, Inc.

»And finally the personal touch assured us we are making a right choice – I mean the proof of concept-workshop we got in June. It left us impressed by their knowledge and professionalism.«

Freenet AG

»Besides the technology, Camunda‘s professional support is also very recommendable.«

Count + Care GmbH

»The support is highly recommendable: Our questions and bug reports are always promptly analyzed, resolved and answered.«

dwp Bank AG

»Especially our experience how Camunda met our individual needs and wishes, was very positive (consulting, finding appointments for bug fixing, or contract terms).«

Consulting and Training

In addition to the support, the available consulting services also form a decision making factor. In this respect, Camunda distinguishes itself with 3 features:

- As our roots lie in consultancy, we feel strongly about the high quality of consultancy services.
- Generally we qualify our customers to help themselves: It is not our strategy to conduct extensive consulting projects, but to enable our customers to find solutions for themselves.
- If you still need extensive external resources, we have a hand-picked network of selected IT service providers based in Europe and North America. These certified partners enjoy our full confidence to provide you with advice in a quality for which we can vouch for.

Feel free to contact our references who have already experienced our consulting:

Zalando AG

»Thanks to Camunda's additional advice we were able to instantly involve all stakeholders into the process design and were also able to swiftly start the technical implementation.«

VHV Insurance

»Last but not least the competent care provided by Camunda's employees as part of a recent workshop could also convince us of a collaboration.«

LVM Insurance

»It only took a few days to highly inspire the whole project team (consisting of people from both IT and business departments) for process mapping with BPMN 2.0, so right now the first improved processes are already emerging.«

Wüstenrot Bausparkasse

»Furthermore, the whole package was very attractive. With Camunda the BPMN knowledge and the needed software come from the same vendor. This made an optimal start with BPMN 2.0 and process automation possible.«

Pricing

Both Camunda BPM and JBoss jBPM are published under open source licenses and thus available for free. However, a commercial version is also offered in the form of a subscription for both products ("Camunda BPM Enterprise Edition" and "JBoss BPM-Suite").

The pricing model for the JBoss BPM Suite subscription is described on 15 pages in a general "Subscription Guide for Red Hat JBoss Middleware".[6] It applies to the BPM suite as well as 10 other products and can therefore not take particular account of the conditions of a BPM platform. The general metric is therefore the number of CPU cores, which are allocated for the use. This includes both the productive use, as well as the use in test and QA environments. Use for development purposes is not at charged for, however only to a limited extent: "Covered by free developer use; entitles up to 25 users per 16 cores of subscription" (page 4).

Some difficulties arise from this model. How can, for example, a process application be evaluated, that embeds the process engine as a resource and requires 16 CPU cores or more, but produces only a small part of its workload due to the process engine?

Camunda BPM is offered under a different pricing model that specifically accounts to the circumstances of an embeddable BPM platform: The number of generated flow node instances per month are deciding. A flow node according to BPMN specification is an event, an activity or a gateway. Therefore, if a very simple process is frequently executed, or a complex process very rarely, the intensity of the use of Camunda (and the derived benefits) are quite comparable. Our pricing model takes this into account, and allows for a very low cost entry. Only the productive use will be counted: Use for testing and QA purposes does not lead to increased costs. The use for development purposes is, unlike the JBoss

[6] https://www.redhat.com/en/files/resources/en-rhjb-subscription-guide-12149557.pdf

BPM suite, also at no extra costs and without any restrictions.

Conclusion

In this document we have shown the – in our opinion – relevant reasons why Camunda BPM is preferable in comparison to JBoss jBPM 7. In fact, there are many other reasons that are beyond the scope of this document. We therefore strongly advised to directly evaluate both products and to check both technical aspects as well as aspects of cooperation with the vendor.

Imprint

Europe / Asia

Camunda Services GmbH
Zossener Str. 55
10961 Berlin
Germany

Phone: +49 (0) 30 664 04 09 - 00
E-Mail: info@camunda.com
www.camunda.de

America

Camunda Inc.
44 Montgomery St, Suite 400
San Francisco, CA 94104
USA

Phone: +1.415.548.0166
E-Mail: info@camunda.com
www.camunda.com