Project no.	609551
Project acronym:	SyncFree
Project title:	*Large-scale computation without synchronisation*

European Seventh Framework Programme
ICT call 10

Deliverable reference number and title:	D.5.1 Tooling: Tools for deployment, benchmarking and testing.
Due date of deliverable:	1 October 2014
Actual submission date:	30 September 2015
Start date of project:	1 October 2013
Duration:	36 months
Name and organisation of lead editor for this deliverable:	Basho Technologies Ltd.
Revision:	1.1
Dissemination level:	CO

Contents

1 Executive summary

The SyncFree project aims for enabling trustworthy large-scale distributed applications in geo-replicated settings. The core concept are replicated yet consistent data types (CRDTs) which allow information dissemination and sharing without the need for global synchronization.

Within the project, Work Package 5 (WP5) coordinates the work to validate the theoretical work on CRDTs on real-life use cases. In the first year of the project, we focused on building out repeatable testing and benchmarking utilities for both the deliverables of Work Package 2 (WP2) and Work Package 4 (WP4). These utilities allow us to test our software under the scenario outlined in the DoW: a static topology with rare failures and limited replicas, but is also modularized enough to allow later extension to testing under dynamic membership and failure conditions.

In the second year of the project

We discuss four major contributions:

Basho Bench We adapt the open-source, Erlang-based, benchmarking utility, *basho_bench* for use with the Work Package 2 (WP2) deliverable, **Antidote**. We discuss this implementation in Section 6.

Riak Test We adapt the open-source, Erlang-based, testing utility, *riak_test* for use with both Work Package 2 (WP2) and Work Package 4 (WP4) deliverables: respectively, **Antidote** and **Derflow**. We discuss this implementation in Section 7.

QuickCheck We use the property-based testing tool for Erlang to build models for the core of the Work Package 4 (WP4) deliverable, **Derflow**. We discuss this implementation in Section 8.

Adaptive Replication We provide a tool for simulation of the adaptive replication algorithm, as discussed in the work of Work Package 1 (WP1) and Work Package 2 (WP2). We discuss this implementation in Section 9.

Antidote A number of test cases and utilities for deployment, measuring and monitoring of Antidote has been added through year2.

2 Milestones in the Deliverable

Work Package 5 (WP5) was not involved in Milestone 1 (MS1).

Tasks delivered during the M12 period were:

Task no	Task name	Date due	Actual date	Lead contractor
T5.1	Tooling: Tools for deployment, benchmarking, replay, and logging and tracing at extreme scale	M18	M18	BASHO

The work on Task 5.1 started in month 6 and continued until month 18, focusing on the following goals, as stated in the project proposal:

The objective of this task is to evaluate the behavior and performance of the outputs from WP 2, WP 3, and WP 4, at scale and under stress from load and various failure scenarios. To do this, we will develop a set of tools for deploying and controlling applications, generating load, simulating failures, capturing and replaying (non-personal) traces for comparison, and logging and analysing outputs and statistics, in a controlled manner. This environment is associated with the platform developed in WP 2 and benefits from the operational expertise of Rovio and Basho in automating deployment and accurately tracing and recording behaviour of the systems. We plan to make use of their expertise and software (Basho's basho_bench and riak_test, Rovio's Skynest Cloud) for orchestrating repeatable automated deployments and tests at scale.

The rest of the document describes to what extent the progress of this task has been achieved, resulting in the work delivered for D5.1.

2.1 M24 Update

The use cases described by Rovio and Trifork in D1.1 where all problems which they had implemented, or tried to implement, using pre-CRDT technology. At the time they felt that CRDTs would help with either correctness or performance and availability.

3 Contractors contributing to the Deliverable

The following contractors contributed to the deliverables:

3.1 Basho

Christopher Meiklejohn (together with WP2 and WP4).
Russell Brown.

3.2 Trifork

Amadeo Ascó (together with WP1).

3.3 INRIA

Tyler Crain

3.4 UNL

Diogo Serra

3.5 UCL

Zhongmiao Li.
Manuel Bravo.

3.6 KL

Annette Bieniusa.
Deepthi Devaki Akkoorath.

4 Background

In this section, we briefly describe the Work Package 2 (WP2) and Work Package 4 (WP4) deliverables: **Antidote** and **Derflow**.

Antidote Antidote is a causally consistent, geo-replicated CRDT data store which features scalable, conflict-free implementations of transactions, by providing consistent, stable snapshots and atomic multi-CRDT updates. In the current version, each data center fully replicates the CRDT store using a static partitioning scheme. Updates are propagated by shipping operations in a causally consistent manner for each transaction. Our approach requires lower bandwidth compared to state-of-the-art architectures. Transactions with Causal+ Consistency semantics support the programmer in observing consistent snapshots of the data without requiring global synchronization.

The source code for Antidote is available on GitHub. `https://github.com/syncfree/antidote`.

Derflow Derflow provides a distributed deterministic dataflow programming model over single-assignment variables and CRDTs. Derflow can run independently or as a component of Antidote. To support this, and ensure correct behavior and preservation of language-level invariants, the core of the programming model has been abstracted from the persistence and data storage layers.

The source code for Derflow is available on GitHub. `http://www.github.com/SyncFree/derflow`.

5 Goals and KPIs

The overall objective is to show that CRDTs are a good thing. We will evaluate that claim in two ways:

- Easier to use — a qualitative assessment for most parts.
- Performance — latency and throughput.

The metrics for ease of use include number of bugs, size of the code and other indicators of that type, to assess the claim.

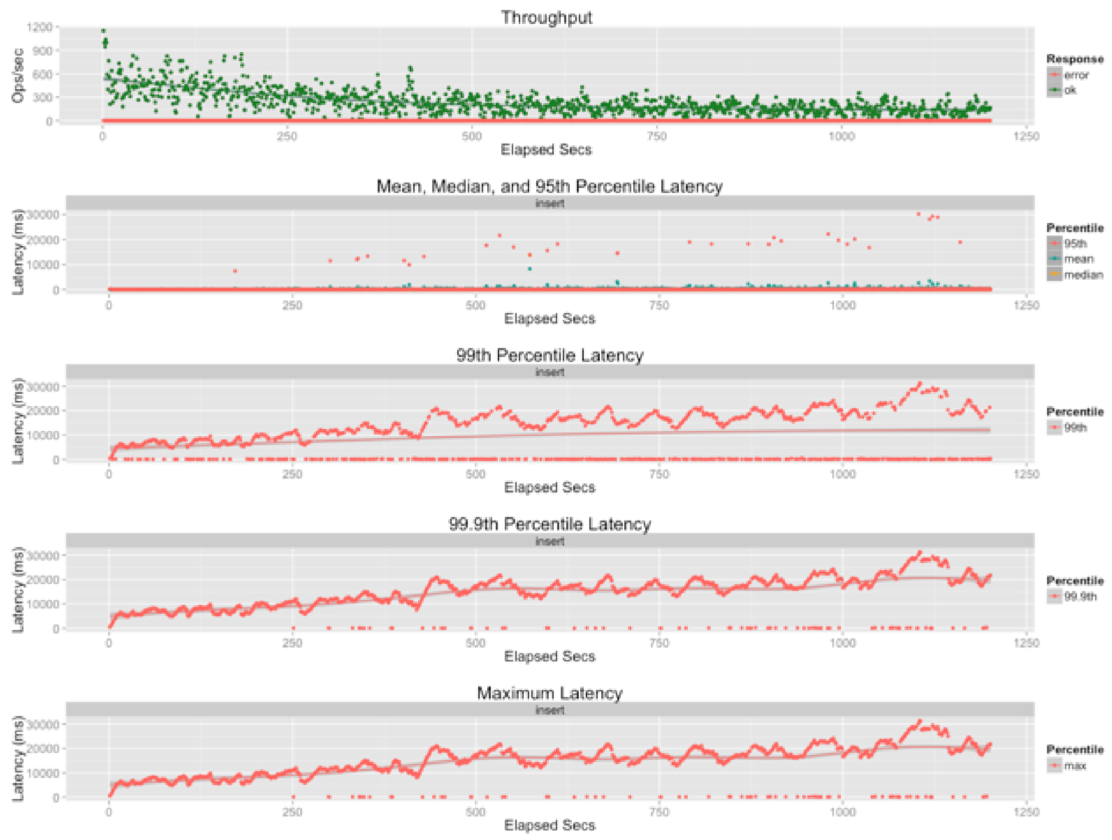
6 Basho Bench

Basho Bench is an open-source, Erlang-based benchmarking tool developed by Basho Technologies, primarily build for testing the operation of Basho's commercial database product, Riak.

Basho Bench was extended during the work for Deliverable D.5.1 to operate with the Work Package 2 (WP2) reference platform *Antidote*. In extending Basho Bench, we adapted the benchmarking tool to operate using the transactional API provided by Antidote. This tool has been used during the development of Antidote to assist in qualifying patches and improvements to the core functionality of the reference platform.

The extension to Basho Bench was done as a driver for Antidote using the driver architecture of Basho Bench. Protocol buffers support were added in a separate driver. On top of that a number of small changes were made to make Basho Bench more generic.

Basho Bench will be used to measure performance using graphs like this one:



7 Riak Test

Both Antidote and Derflow are built on top of the Erlang-based distributed systems framework, *riak_core*. Riak Core provides a series of essential components for building distributed, fault-tolerant, highly-available applications. These include, but are not limited to, consistent hashing and managing a group of nodes with dynamic membership. Riak Core was abstracted out of the distributed database

Riak, to provide a basis for building other applications.

Riak Test, the testing tool developed for Riak by Basho Technologies, is primarily focused on testing of Riak clusters. During the work for Deliverable D.5.1, we identified and performed a series of modification to the Riak Test library, which were contributed back to the mainline, to support operations over Riak Core application, instead of only Riak itself.

In the process of this work, we developed a series of tests as well as a streamlined testing process for qualifying the behavior of both *Antidote* and *Derflow (aka LASP).*

A selection of some of the tests added to validate Antidote:

append_failures_test Partitions the network while writing objects to the log and verifies that the log is merged correctly when the partition heals.

append_test Verifies that operations can be written and read back from the log.

clocksi_test Verifies the Clock SI protocol implementation: certification check, multiple read/writes, concurrency.

inter_dc_repl_test Verifies the functionality of the inter-DC extension for Clock SI.

log_handoff_test Verifies that during an hand-off process updates are transferred between nodes.

log_test Basic test of the log: perform a sequence of write operations to a counter and ensure that the correct value for the counter is read back.

mvreg_test Verifies the implementation of a MV-Register CRDT.

opbcounter_test Verification of a Bounded Counter CRDT.

oporset_test Verifies the implementation of the operation-based OR-set.

pb_client_test Verifies that the protocol buffer API functions correctly.

The streamlined testing process builds releases and automatically runs the test suite for every test in the riak_test directory.

For LASP, there is a test case for *every* feature in the language. A single 'make' target will automatically download riak_test, configure it, build a bunch of development releases, and run the entire suite. This enables using Travis CI for testing all of the riak_test work.

These tests are not only used for qualifying new changes to both applications, but also are used for ensuring scalability and performance improvements do not degrade or harm performance of the current system.

8 QuickCheck model for core of Derflow

The Work Package 4 (WP4) deliverable, *Derflow* (aka LASP), provides a programming model that supports distributed deterministic dataflow programming. This model relies on a variable store, which is shared between all processes performing computations.

Derflow can be configured to run in different environments:

Standalone In standalone model, Derflow can run programs using its own distribution model, which stores variables used in computations in the Erlang Term Storage system.

Antidote Derflow is also aimed at running on top of the Antidote reference platform for applications requiring Causal+ Consistency using Antidote's storage layer as the backing variable store.

Riak Derflow can also run on top of Riak, for use in testing applications deployed at scale in production, such as our industry partners from Rovio.

To this effect, we have abstracted the core language semantics and invariants out from the variable storage layer. In performing this abstraction, we have developed a QuickCheck model that will be used to validate correct execution of new backend implementations of the programming model, as we optimize performance.

9 Adaptive Replication Simulation Tool

In the context of Deliverable 2.2 of Work Package 2 (WP2), we have studied CRDTs in partial-replication settings. In one line of this work, we have investigated dynamic strategies for CRDTs replication on a subset of nodes such that replicas reside close to users. This work is discussed in detail in Deliverable 2.2.

In the context of Work Package 5, we developed a simulator which implements the adaptive replication algorithm and allows us to test it as it is being refined. In particular, the simulator allows us to investigate the influence of different parameter settings, such as replication factors, varying loads, and threshold values for allocating or moving replicas. This tool provides an interactive help utility, a "log view" which displays a log of operations as they occur during execution of the algorithm, as well as a "graphical view" which shows how the replicas are being moved based on the algorithm's output.

The code can be found in the SyncFree github repositories (`https://github.com/SyncFree/AdaptiveReplicationTool`).

10 Megaload

One major benefit of bringing Erlang Solutions into the consortium is that we can utilise the Megaload tool. Megaload was initially developed in the Prowess EU FP7 project and is now being taken to market by ESL.

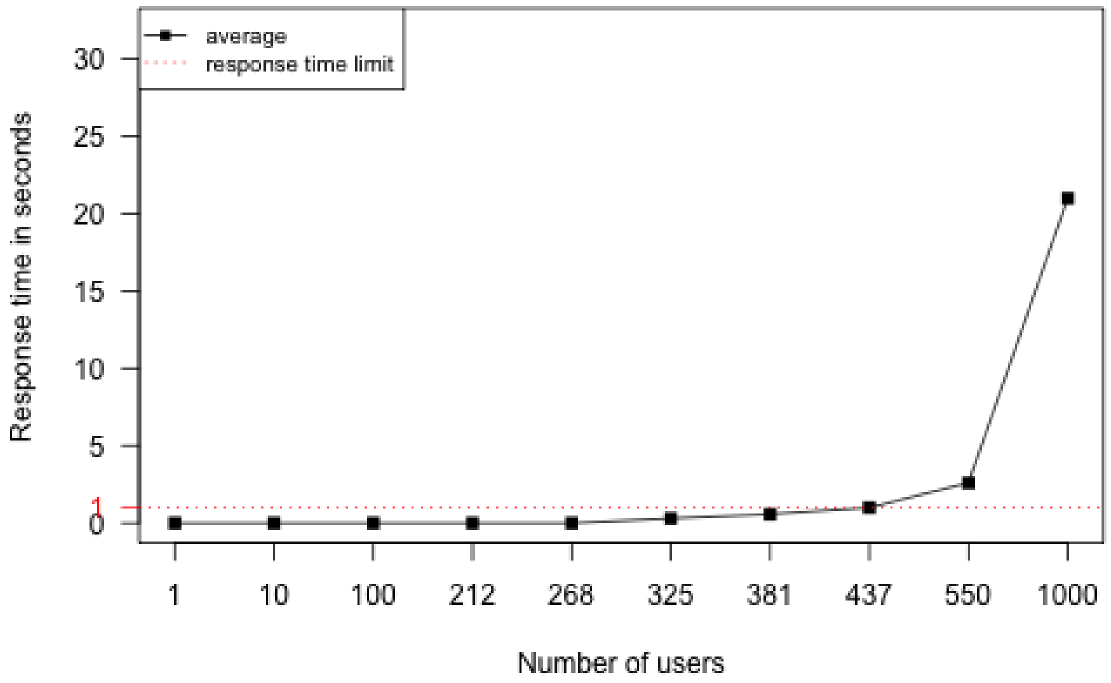
From the Megaload flyer (`http://www.prowessproject.eu/wp-content/uploads/2015/06/Megaload-Flyer.pdf`):

> Megaload is a scalable load testing tool that provides automatic deployment on cloud environments or physical hardware, allowing you to simulate a massive amount of load to stress test your system. The powerful real time measurement system provides all the information you need to monitor your tests through the graphical user interface. Megaload is ideal for online business, SaaS and telecoms companies.

A Megaload test is described and parameterised by a JSON script. A test is a container of phases, which decide how many users are simulated, at which rate are they started and how many requests per second are generated. Several phases can combined into a sequence provided to the *test* object. The users simulated are described by the *scenario* object.

Megaload will be used to drive the test in the large scale experimentation (D5.4). For generating realistic workloads, we will use basically two types of information: generic information of the ratio of operation in common workloads provided by Rovio; information extracted from analysis real logs provided by Basho for some specific services.

For the large scale testing of Antidote we will also use the graphs generated by Megaload, e.g., response times as below.



No work on Megaload itself is expected in SyncFree, but test cases will be developed in order to exercise the system under test. This tool will also allow us to replace all ad-hoc software developed to test the different prototypes produced in the other Work Packages.

11 Plans for M24-M36

In addition to the load testing we will also investigate failure injection using the Commander tool developed by Koç.

The Commander tool provides prioritized systematic exploration of different event interleavings in a multi-DC system. This is tantamount to exploring the injection of faults in the form of inter-DC message delays, re-orderings and/or losses. We start with a recorded execution of the multi-DC system and gradually insert faults. In each of the fault-injected executions, we determine whether the fault(s) result in violations of replica-local assertions and invariants on final global state. Using the Commander tool, application developers can explore how resilient their CRDT-based programs are to such faults, and introduce invariant preservation, recovery and repair mechanisms to their programs as needed.

12 Choice of use cases

12.1 Wallet

Rovio had a Wallet in development, and planned to use conflict resolution based on vector clocks for wallet balance in order to guarantee correctness in all cases and even with overlapping writes to same variable. Their experience shows that ad-hoc and custom conflict resolution is error prone and difficult to implement right (a fair share of sibling explosions, et cetera, in the wallet proves the point) and thus a more standardized and easy-to-use way would have been useful to reduce implementation complexity and improve operational robustness.

CRDTs were seen as the ideal way to implement robust and standard conflict resolution for the wallet use case without extra implementation overhead. CRDTs are standardized data types with well defined behaviour in conflict cases. Due to the standardization it would also possible to implement CRDTs on the database side (and not in the client library or as custom conflict resolution handler) for maximum efficiency and make it possible to write code that "Fire and forget" writes to the database, instead of Riak's read+write way which introduced additional overhead and latency to the process.

Because of these reasons Rovio chose Wallet as a real-world use case for the SyncFree project and due to data criticality and fact that Wallet would be in the real operational and business use the large-scale testing was warranted.

However, changes in Rovio's plan (e.g. no loyalty point scheme yet in the wallet) and greatly delayed Riak 2.0 rollout with CRDT data types, made it impossible to do timely testing for Wallet development. Rovio had to rely on custom conflict resolution without CRDTs. Currently Rovio doesn't have the resources nor plans to rewrite the Wallet implementation to use CRDT types and proceed with larger scale testing.

However, the Wallet example has been adopted by the academic partners using Antidote, so the knowledge from Rovio has been passed on to that application. More on this below.

12.2 Leaderboard

For the leaderboards, Rovio has not been able to find a game that would clearly benefit from distributed score-handling over sending scores to a centralized server. Some peer-to-peer games would benefit from this, but in practice Rovio does not have such games in production or plans to prototype at the moment.

There are potentially other applications for CRDT types in online multiplayer worlds (e.g. large scale Minecraft style gaming) where it's inpractical (esp. in mobile networks) to keep the game world completely synchronized between all actors, but currently Rovio doesn't have concrete plans on taking such initiative as it's not needed for Rovio's near-term business.

12.3 Ad Service

Ad Service use case was about making a robust way to count ad impressions that could be utilized for real-time serving logic for different kinds of campaigns. However, it turned out that due to complexities in rules for campaign serving and in balancing inventory between the campaigns a simple counter would not be that useful.

Rovio has since changed the architecture to be based on precalculated serving lists for campaigns as that allows the usage of complex business rules without real-time processing constraints. For the impression reporting it is relying on a big data analytics pipeline. In practice this means that Rovio doesn't currently see long-term need for applications for the distributed counters related to the Ad serving.

12.4 Configuration files

ESL will — as part of D5.3 — compare a non-CRDT solution to a CRDT based solution for the management of configuration files in a huge Erlang cluster.

For customers with hundreds of Erlang nodes running, it is a nightmare to ensure that all nodes are using the same configuration, so an automated synchronisation of them is required to ensure flawless operation of the system. This feature would add a lot to the operational value of the operations and maintenance tool *Wombat* that ESL sells.

Antidote will be used as the CRDT technology to allow more room for experimentation and changes to the CRDTs, should that need arise. This is intended as a simple use case to verify the scalability of Antidote to many nodes and geo-replication.

12.5 Wallet with Antidote

In order to get a big enough test example the Wallet implemented in Antidote will be deployed on hundreds of nodes and tested by Megaload in D5.3. This version of the Wallet has been developed based on the requirements Rovio had to their Wallet and implemented by the academic partners to validate Antidote and show how a solution based on CRDTs would look like.

This will give excellent feedback on how scalable Antidote is.

Megaload will be used to generate the load on the system.

12.6 Big Sets

Basho is working on performance improvements to implement Big Sets (to be reported later in WP2) and for that there will be testing conducted as part of D5.2. If at all possible Basho will find a client to conduct the testing with, but should that fail, a test based on what customers have been asking for will be conducted instead.

13 Use of resources

We have been able to re-use a lot more of the Riak code base than originally anticipated, which means that we have used fewer resources than planned. Additionally, all the work done by non-EU residents from Basho will not be charged to the project, but just donated to it.

13.1 Rovio's situation

During the SyncFree project, Rovio has been in a challenged position as just one year ago (October 2014) it was forced to cut 110 jobs and currently (September 2015), it's again negotiating to further reduce it's workforce with 260 jobs. This has unfortunately affected also to the this project, as Rovia has lost people working in and close to this project. An example of this is the Leaderboard area, which may in the future be something where Rovio will seek to use external service providers, rather than building the service stack itself. Rovio does definitely *not* want to cancel it's participation into the project, but the challenges Rovio has had, have been causing difficulties in the scale of the tasks Rovio can commit itself into.

14 Papers and publications

No publications have directly emerged from the work in WP5. However, the test infrastructure work we have performed for WP5 has been used by Derflow, which was published as part of the WP4 deliverables.

References