Full Automated Continuous Integration and Testing Infrastructure for Maxscale and MariaDB

Mark Zaslavskiy[1,3], Alexander Kaluzhniy[1,2], Tatyana Berlenko[1,2], Ilfat Kinyaev[1,2], Kirill Krinkin[1,2]

Timofey Turenko
MariaDB Corporation Ab

[1]FRUCT LLC
[2]Saint-Petersburg State Electrotechnical University «LETI»
[3]ITMO University Saint-Petersburg, Russia

Motivation

Quality assurance procedure for distributed database management systems is time and resource consuming because growing variety of:

- Supported product versions,
- Use cases,
- Environment options (OS and software dependencies versions),
- RDBMS configuration options,
- Network configurations.

Maxscale

- Intelligent DB proxy for the MariaDB;
- Provides increased level of availability, scalability and security for MariaDB/MySQL cluster.

The work aims to provide solution for **automating integration tests and environment management** for Maxscale and backend database.

Testing tasks

- **Build** and publish Maxscale binary packages (rpm and deb).
- **Run and test.** Install Maxscale packages on a clean virtual environment and execute integration tests.
- **Upgrade test**. Install new Maxscale version on a virtual machine with old one and validate update.

Previous solution

- A set of Bash scripts with hardcoded parameters (IPs, VM number).
- Virtualization providers: Libvirt/QEMU and Amazon Web Services
- Limited error processing.
- Limited results processing.
- Single ssh key for all test runs.

Previous infrastructure - libvirt/QEMU

- Sequence of procedures for preparing machines:
  - preconfigure image;
  - copy image;
  - bring up machines one by one;
  - config each machine network.
- Disadvantages:
  - VM preparation is slow.
  - Network configuration is done by script.

Previous infrastructure - AWS

- Algorithm
    - Bring up 9 AWS instances.
    - Gather AWS machines data.
- Disadvantages:
    - Manual cleanup after tests.

Solution requirements

- Provide reliable and easy extendable test automation instrument.
- Support variety of:
  - virtualization providers,
  - OS,
  - products.
- VM management:
  - reliability,
  - automatic network configuration,
  - smart cleanup (collect artifacts and destroy VMs).

Proposed solution - MDBCI

MDBCI (MariaDB Continuous Interface) -

- a set of tools for testing MariaDb and Maxscale,
- domain-specific Ruby wrapper around Vagrant.

The main features of MDBCI:

- Support of Docker, Libvirt, VirtualBox, AWS and PPC.
- Automatic and robust creation of VM set by configuration template.
- Automatic deploy MariaDb/Galera/Maxscale to VM nodes, running configuration procedures.
- Products repository management on all available platforms.

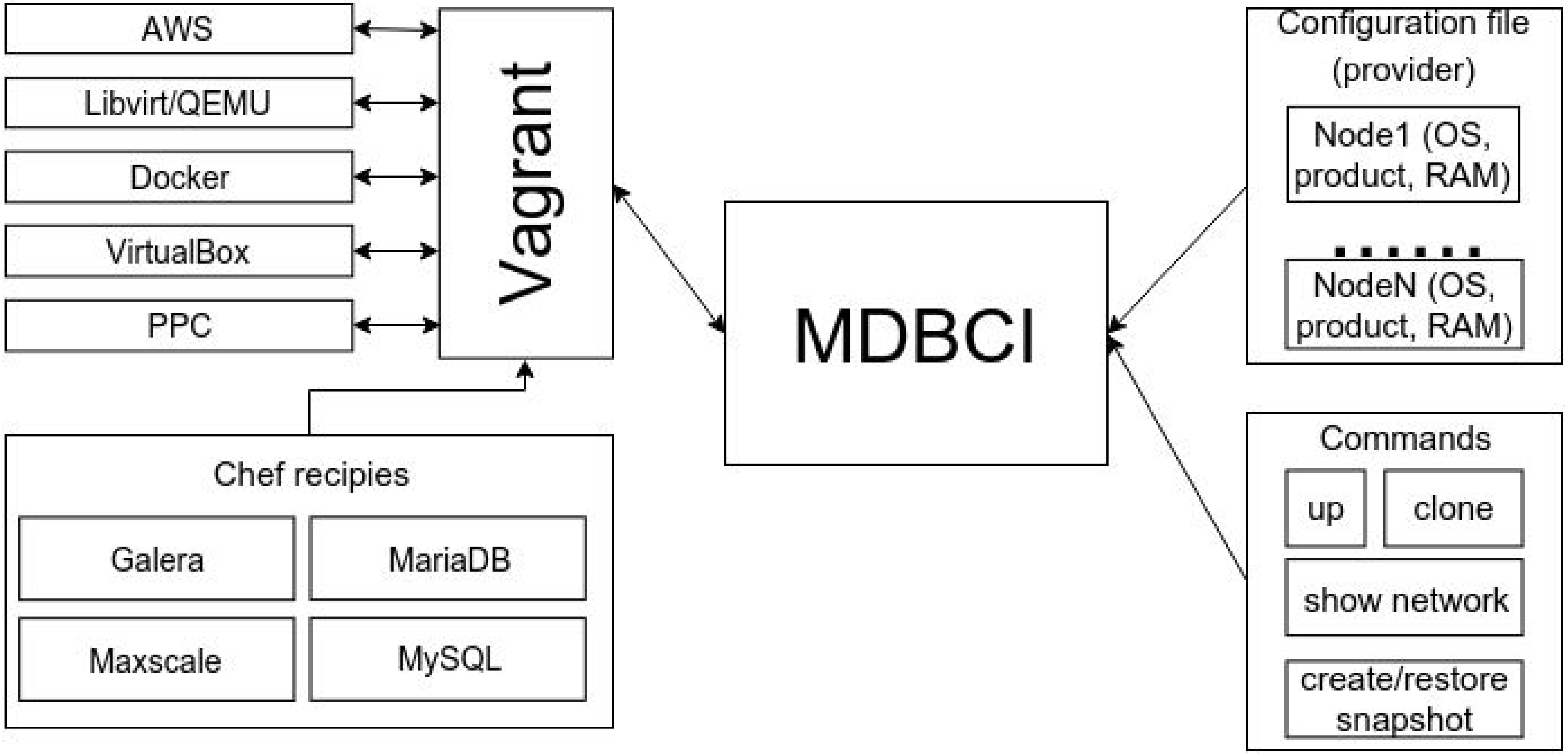
MDBCI additional features

- Parsing and storing of performance (sysbench) and regression (CTest) testing results.
  - Deep integration with Jenkins testing jobs;
  - All test runs are processed and stored into special database, which allows to compare different product commits/settings.
- Snapshots and cloning interfaces for Docker and Libvirt configurations.

Configuration launch usecase

1. Create a config file.
2. Download needed Vagrant boxes.
3. Generate Vagrant config file by MDBCI config.
4. Run configuration using MDBCI.
5. After successful launch of configuration all VM network information is exported to file.
6. Further VM control can be done with MDBCI or Vagrant.
7. (Optional) Create clone of currently running VMs.
8. (Optional) Create snapshot of currently running VMs.

Architecture

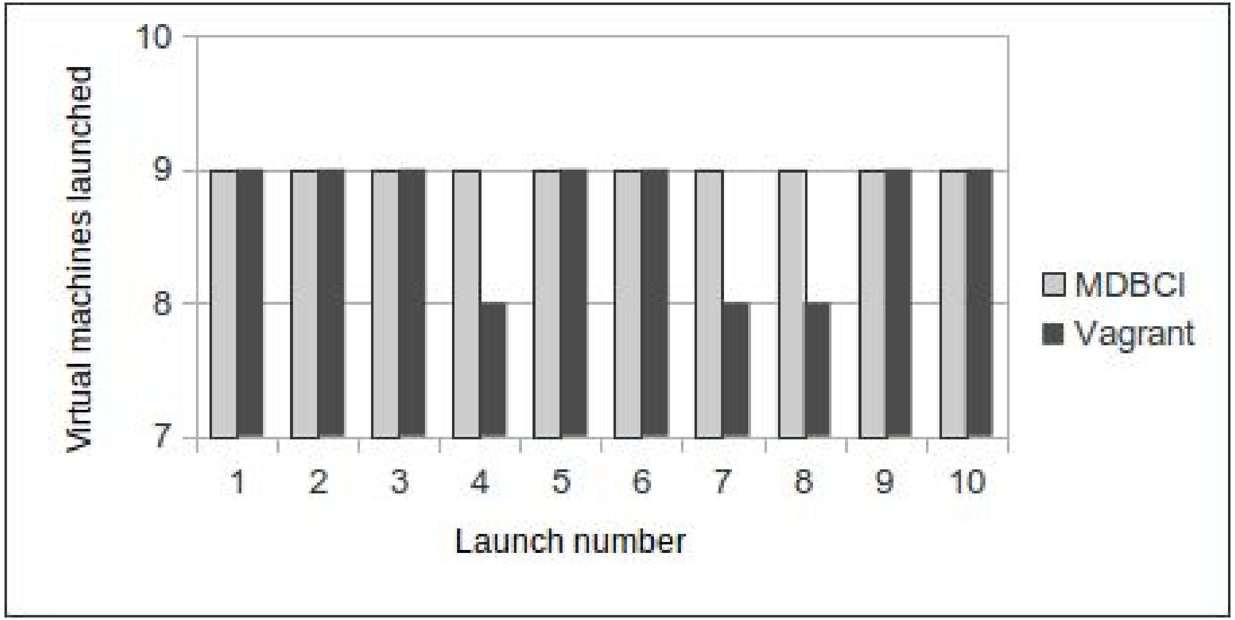


Evaluation

- Goal - measure VM launch fails ratio on AWS configuration.
- Experiments steps
  1. Test configuration using MDBCI command generate.
  2. 10 sequential launches of the generated configuration using “vagrant up” command.
  3. 10 sequential launches of the generated configuration using “mdbci up” command.
  4. Calculation of failed launches percentage for each instrument.

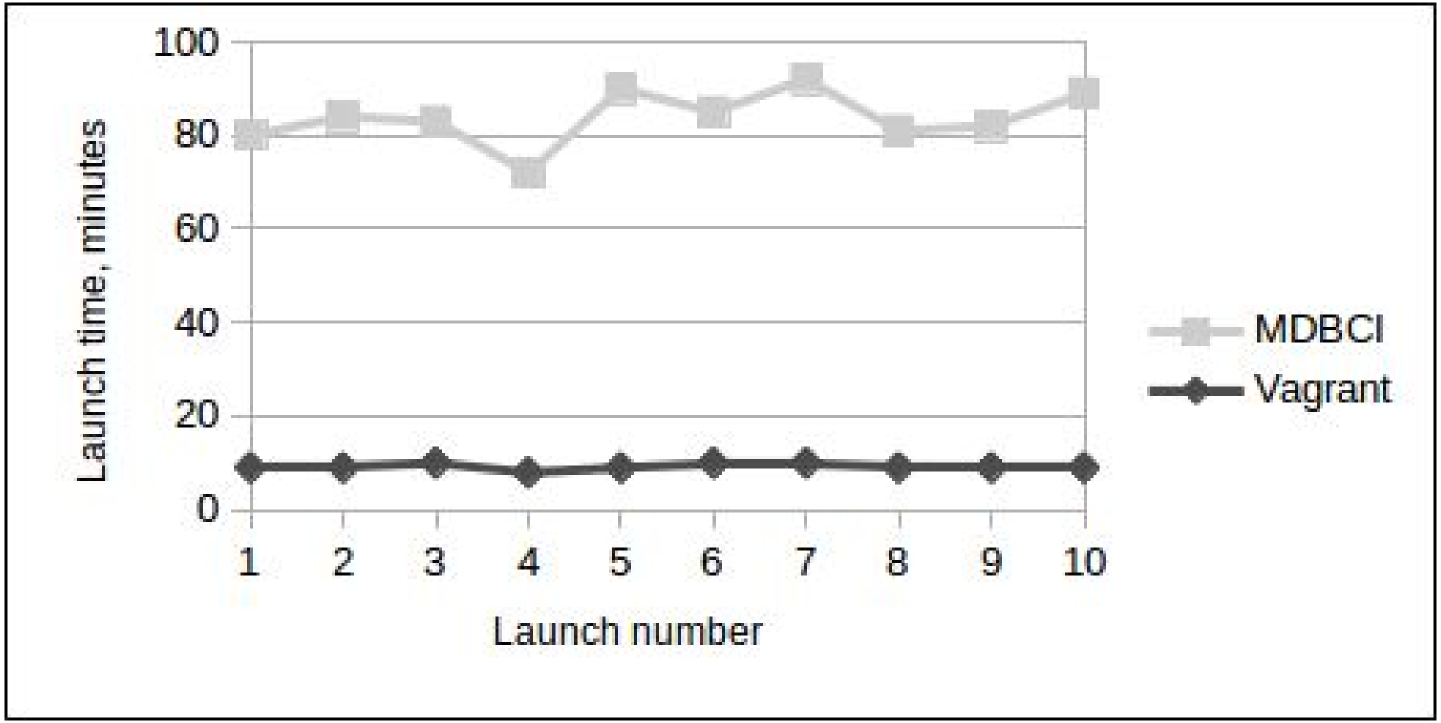
Evaluation

- Number of successfully running machines after per test configuration launch



Evaluation

- Time of test configuration launches



Conclusion

- MDBCI allowed to overcome limitations of the previous test automation: nontransparent debug interfaces, limited support of virtualization providers.
- MDBCI fully integrated to Maxscale development process.
- Future development of MDBCI includes
    - embedding new virtualization providers,
    - different performance benchmarks support,
    - interfaces for multi provider configurations are planned.

Links

Questions: mark.zaslavskiy@fruct.org

Source

- https://github.com/OSLL/mdbci

Maxscale

- https://github.com/mariadb-corporation/maxscale
- https://mariadb.com/products/mariadb-maxscale

Configuration examples

```
{
 "galera1" :
  {
   "hostname" : "galera1",
   "box" : "ubuntu_trusty_libvirt",
   "product" : {
     "name": "galera",
     "version": "5.5",
     "cnf_template" : "server1.cnf",
     "cnf_template_path":
"/home/vagrant/cnf"
    }
  },
```

```
 "maxscale" :
  {
   "hostname" : "maxscale",
   "box" : "ubuntu_trusty_libvirt",
   "product" : {
     "name" : "maxscale"
   },
   "memory_size" : "1024"
  }
}
```