# Introduce Chaos Experiments in Apache Spark

Author: Qian Sun
Shepherd: Hyukjin Kwon

**Q1. What are you trying to do? Articulate your objectives using absolutely no jargon.**

Introduce chaos experiments in apache spark to make sure spark can withstand the unpredictability of in-production environment, to help developers more quickly identify and resolve issues that might not be captured with unit and integration testing.

**Q2. What problem is this proposal NOT designed to solve?**

- NOT designed to ensure all code logic tested.
- NOT designed to guarantee that spark can work well with other components.
- NOT designed to make sure spark performance.

**Q3. How is it done today, and what are the limits of current practice?**

No chaos experiments executed in apache spark.

**Q4. What is new in your approach and why do you think it will be successful?**

There are several distributed systems that execute chaos experiments, such as apache apisix, apache pulsar, tidb, RabbitMQ, tikv.

Distributed systems face complex and unpredictable production environments, such as disk failures, machine power loss, network isolation, and that's just the tip of the iceberg. To make distributed systems  more robust, we need a method to simulate unpredictable failures and test responses to these failures.

After chaos experiments:

- Increases reliability and resiliency for apache spark.
- Unplanned downtime and outages are far less likely to occur due to proactive and constant testing.
- Strengthens system integrity.

**Q5. Who cares? If you are successful, what difference will it make?**

Anyone who deploys apache spark in production environment cares. It will help apache spark to expose issues about reliability and resiliency faster and earlier, making it easier to reproduce user-reported production issues.

**Q6. What are the risks?**

Technically, there are no more risks yet given that there are many distributed systems already supported.

At present, it has been verified by github action CI. During working, the chaos experiment will be implemented through code, while supporting local chaos experiments and GA.

**Q7. How long will it take?**

Before Apache Spark v3.4.0, it will take about 3-4 month to complete after SPIP approved

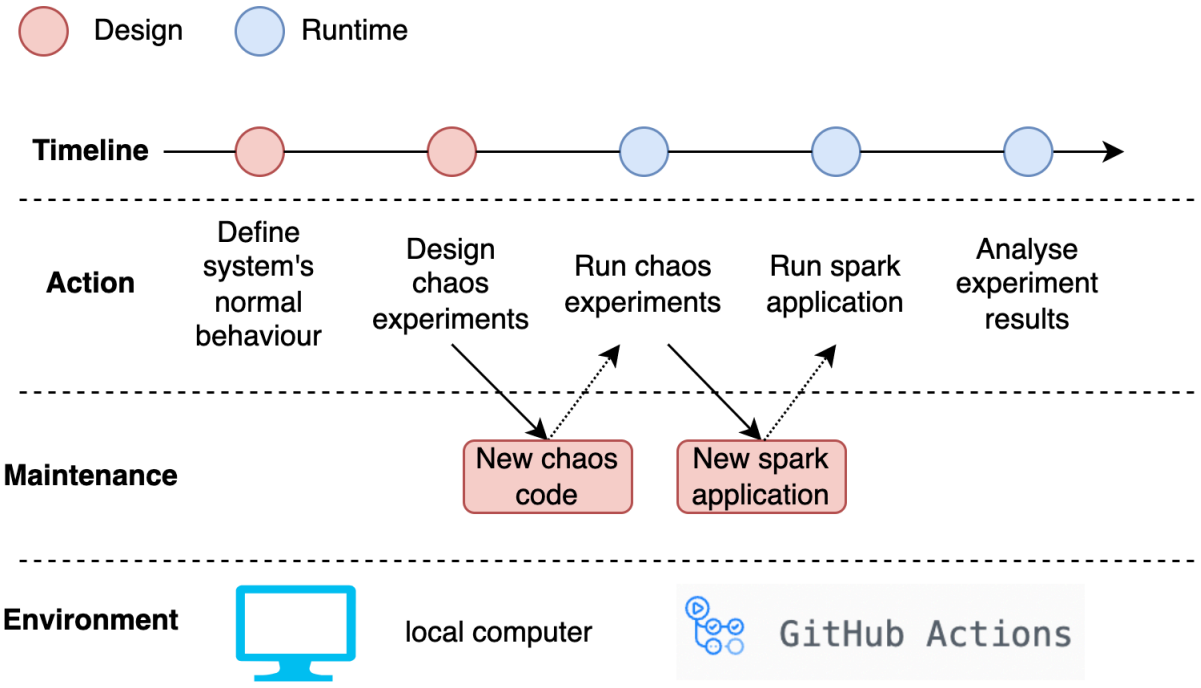
**Q8. What are the mid-term and final "exams" to check for success?**

- Chaos experiments can be performed locally via script or mvn command in mid-term.
- Chaos experiments are integrated in github action to validate each PR in final.

**Appendix A. Proposed API Changes. Optional section defining APIs changes, if any. Backward and forward compatibility must be taken into account.**

No API changed because new chaos experiment code is in the test folder.

**Appendix B. Optional Design Sketch: How are the goals going to be accomplished? Give sufficient technical detail to allow a contributor to judge whether it's likely to be feasible. Note that this is not a full design document.**




1. **Define system's normal behavior**

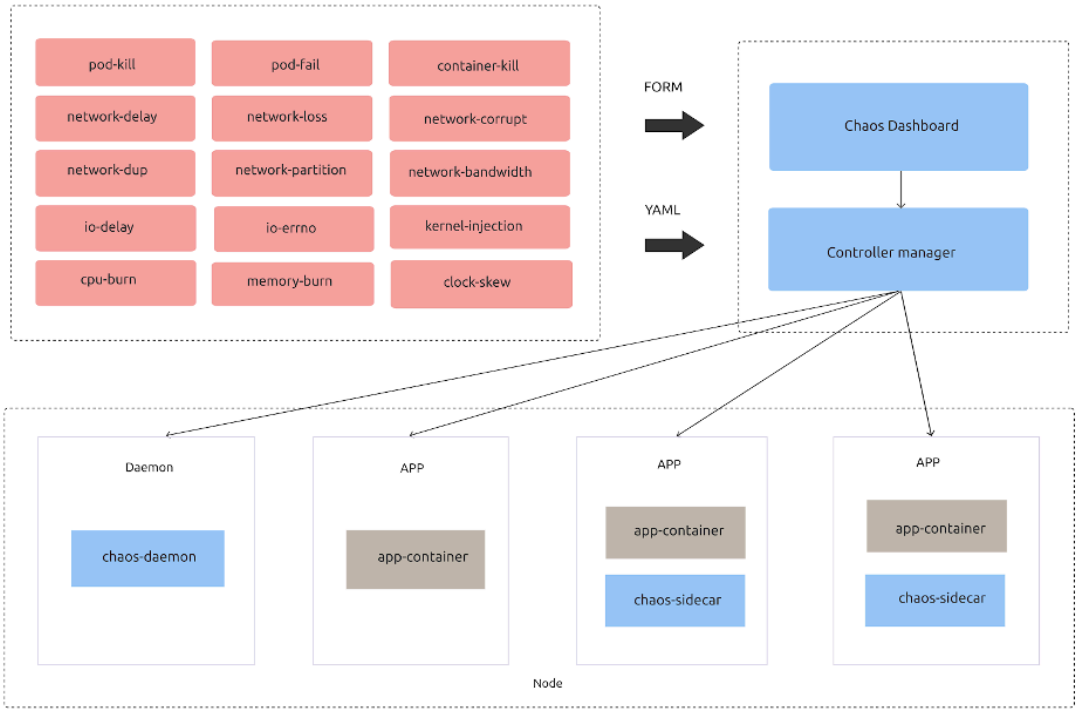
   The system's normal behavior is believed to be acceptable behavior and unexpected behavior. For apache spark, we could validate normal behavior by spark application end state.

   Example: IT for spark on k8s

## 2. Design chaos experiments

Chaos Mesh could be used as a chaos experiment platform, which offers various types of fault simulation and has an enormous capability to orchestrate fault scenarios. Chaos Mesh is built on k8s CRD, primarily contains components:

- **Chao Controller Manager**: The core logical component of Chaos Mesh. Chaos Controller Manager is primarily responsible for the scheduling and management of Chaos experiments. This component contains several CRD Controllers, such as Workflow Controller, Scheduler Controller, and Controllers of various fault types.
- **Chaos Daemon**: The main executive component. Chaos Daemon runs in the DaemonSet mode and has the Privileged permission by default (which can be disabled). This component mainly interferes with specific network devices, file systems, kernels by hacking into the target Pod Namespace.




Currently, the fabric8io provide chaosmesh as extension, so apache spark could schedule chaos experiments through fabric8 kubernetes-client. The benefits of doing this are as follows:

- No new components are introduced.
- Chaos Mesh version can be upgraded with fabric8io version.

*Note that experiments described below are scheduling tasks, which can automatically create chaos at a fixed time, and we can specify the duration of chaos.*

***PodChaos –*** To simulate fault scenarios of the specified Pods or containers, such as pod failure, pod kill, container kill.

| Component | Function | Expected behavior |
|---|---|---|
| spark executor | Running individual tasks by being in charge of a given spark application. | Spark application succeeds. |
| kube-apiserver | Validate and configure data for the api objects which include pods, services, replicationcontrollers, and others. | Spark application succeeds. |
| kube-controller-manager | A daemon that embeds the core control loops shipped with k8s. | Spark application succeeds. |
| kube-scheduler | Control plane process which assigns Pods to Nodes. | Spark application succeeds. |

***NetworkChaos –*** To simulate a network fault scenario for a cluster, such as disconnection, high delays, low bandwidth.

| Component | Function | Expected behavior |
|---|---|---|
| spark driver/executor | Driver: Maintaining information about the Spark Application; responding to a user's program or input; and analyzing, distributing, and scheduling work across the executors (defined momentarily).<br><br>Executor: Running individual tasks by being in charge of a given spark application. | Spark application succeeds. |
| kube-apiserver | Validate and configure data for the api objects which include pods, services, replicationcontrollers, and others. | Spark application succeeds. |

***IOChaos –*** To simulate a scenario of file system fault, such as latency, fault.

| Component | Function | Expected behavior |
|---|---|---|
| spark driver/executor | Driver: Maintaining information about the Spark Application; responding to a user's program or input; and analyzing, | Spark application succeeds. |

| | | |
|---|---|---|
| | distributing, and scheduling work across the executors (defined momentarily).<br><br>Executor: Running individual tasks by being in charge of a given spark application. | |

***TimeChaos*** – To simulate a time offset scenario.

| Component | Function | Expected behavior |
|---|---|---|
| spark driver | Maintaining information about the Spark Application; responding to a user's program or input; and analyzing, distributing, and scheduling work across the executors (defined momentarily). | Spark application succeeds. |
| spark executor | Running individual tasks by being in charge of a given spark application. | Spark application succeeds. |

***StressChaos*** - To simulate stress scenarios inside containers, such as cpu stress, memory stress.

| Component | Function | Expected behavior |
|---|---|---|
| spark driver | Maintaining information about the Spark Application; responding to a user's program or input; and analyzing, distributing, and scheduling work across the executors (defined momentarily). | Spark application succeeds. |
| spark executor | Running individual tasks by being in charge of a given spark application. | Spark application succeeds. |

3. **Run chaos experiments**

   Run chaos experiments with arguments

| Type | script arguments | maven properties | Description |
|---|---|---|---|
| PodChaos | podchaos-schedule | spark.chaos.pod.schedule | The time when a pod chaos experiment occurs. For example, '5 * * * *' means experiment occurs every 5 mins. |

| | | | |
|---|---|---|---|
| | podchaos-duration | spark.chaos.pod.duration | The duration of pod chaos. |
| | podchaos-limit | spark.chaos.pod.limit | The number of reserved pod chaos experiments. When there are more than *historyLimit* tasks, earliest created experiments will be deleted. |
| | podchaos-concurrency | spark.chaos.pod.concurrency | Used to specify whether to allow the creation of multiple concurrent pod chaos experiments.<br>● Forbid: multiple experiments are not allowed to be created simultaneously.<br>● Allow: multiple experiments are allowed to be created simultaneously. |
| | podchaos-mode | spark.chaos.pod.mode | Specify the mode of pod chaos experiment:<br>● one: selecting a random pod<br>● all: selecting all eligible pods<br>● fixed-percent: selecting a specified percentage of pods from the eligible pods<br>● random-max-percent: selecting the maximum percentage of pods from the eligible pods |
| | podchaos-range | spark.chaos.pod.range | Provicdes a parameter for the *podchaos-mode* configuration. For example, when *podchaos-mode* is set to *fixed-percent*, *podchaos-range* specifies the percentage of pods. |
| NetworkChaos | netchaos-schedule | spark.chaos.net.schedule | The time when a network chaos experiment occurs. For example, '5 * * * *' means experiment occurs every 5 mins. |
| | netchaos-duration | spark.chaos.net.duration | The duration of network chaos. |
| | netchaos-limit | spark.chaos.net.limit | The number of reserved network chaos experiments. When there are more than *historyLimit* tasks, the earliest created experiments will be deleted. |
| | netchaos-concurrency | spark.chaos.net.concurrency | Used to specify whether to allow the creation of multiple concurrent network chaos experiments.<br>● Forbid: multiple experiments are not allowed to be created simultaneously.<br>● Allow: multiple experiments are allowed to be created simultaneously. |
| | netchaos-mode | spark.chaos.net.mode | Specify the mode of network chaos experiment:<br>● one: selecting a random pod<br>● all: selecting all eligible pods<br>● fixed-percent: selecting a specified percentage of pods from the eligible pods<br>● random-max-percent: selecting the maximum percentage of pods from the eligible pods |

| | | | |
|---|---|---|---|
| | netchaos-range | spark.chaos.net.range | Provides a parameter for the *netchaos-mode* configuration. For example, when *netchaos-mode* is set to *fixed-percent*, *netchaos-range* specifies the percentage of pods. |
| IOChaos | iochaos-schedule | spark.chaos.io.schedule | The time when an io chaos experiment occurs. For example, '5 * * * *' means experiment occurs every 5 mins. |
| | iochaos-duration | spark.chaos.io.duration | The duration of io chaos. |
| | iochaos-limit | spark.chaos.io.limit | The number of reserved io chaos experiments. When there are more than *historyLimit* tasks, the earliest created experiments will be deleted. |
| | iochaos-concurrency | spark.chaos.io.concurrency | Used to specify whether to allow the creation of multiple concurrent io chaos experiments.<br>● Forbid: multiple experiments are not allowed to be created simultaneously.<br>● Allow: multiple experiments are allowed to be created simultaneously. |
| | iochaos-mode | spark.chaos.io.mode | Specify the mode of io chaos experiment:<br>● one: selecting a random pod<br>● all: selecting all eligible pods<br>● fixed-percent: selecting a specified percentage of pods from the eligible pods<br>● random-max-percent: selecting the maximum percentage of pods from the eligible pods |
| | iochaos-range | spark.chaos.io.range | Provides a parameter for the *iochaos-mode* configuration. For example, when *iochaos-mode* is set to *fixed-percent*, *iochaos-range* specifies the percentage of pods. |
| | iochaos-delay | spark.chaos.io.delay | Specify the delay time of io chaos. |
| TimeChaos | timechaos-schedule | spark.chaos.time.schedule | The time when a time chaos experiment occurs. For example, '5 * * * *' means experiment occurs every 5 mins. |
| | timechaos-duration | spark.chaos.time.duration | The duration of time chaos. |
| | timechaos-limit | spark.chaos.time.limit | The number of reserved time chaos experiments. When there are more than *historyLimit* tasks, the earliest created experiments will be deleted. |
| | timechaos-concurrency | spark.chaos.time.concurrency | Used to specify whether to allow the creation of multiple concurrent io chaos experiments.<br>● Forbid: multiple experiments are not allowed to be created simultaneously. |

| | | | |
|---|---|---|---|
| | | | ● Allow: multiple experiments are allowed to be created simultaneously. |
| | timechaos-mode | spark.chaos.time.mode | Specify the mode of time chaos experiment:<br>● one: selecting a random pod<br>● all: selecting all eligible pods<br>● fixed-percent: selecting a specified percentage of pods from the eligible pods<br>● random-max-percent: selecting the maximum percentage of pods from the eligible pods |
| | timechaos-range | spark.chaos.time.range | Provides a parameter for the *timechaos-mode* configuration. For example, when *timechaos-mode* is set to *fixed-percent*, *timechaos-range* specifies the percentage of pods. |
| | timechaos-offset | spark.chaos.time.offset | Specify the length of time offset. For example, *-5m* means time offset 5 mins backward. |
| StressChaos | stresschaos-schedule | spark.chaos.stress.schedule | The time when a stress chaos experiment occurs. For example, '5 * * * *' means experiment occurs every 5 mins. |
| | stresschaos-duration | spark.chaos.stress.duration | The duration of stress chaos. |
| | stresschaos-limit | spark.chaos.stress.limit | The number of reserved stress chaos experiments. When there are more than *historyLimit* tasks, the earliest created experiments will be deleted. |
| | stresschaos-concurrency | spark.chaos.stress.concurrency | Used to specify whether to allow the creation of multiple concurrent io chaos experiments.<br>● Forbid: multiple experiments are not allowed to be created simultaneously.<br>● Allow: multiple experiments are allowed to be created simultaneously. |
| | stresschaos-mode | spark.chaos.stress.mode | Specify the mode of stress chaos experiment:<br>● one: selecting a random pod<br>● all: selecting all eligible pods<br>● fixed-percent: selecting a specified percentage of pods from the eligible pods<br>● random-max-percent: selecting the maximum percentage of pods from the eligible pods |
| | stresschaos-range | spark.chaos.stress.range | Provides a parameter for the *stresschaos-mode* configuration. For example, when *stresschaos-mode* is set to *fixed-percent*, *stresschaos-range* specifies the percentage of pods. |

| | | | |
|---|---|---|---|
| | stresschaos-memory | spark.chaos.stress.memory | Specify the memory size to be occupied. |
| | stresschaos-load | spark.chaos.stress.load | Specify the percentage of CPU occupied.<br>● 0 means that no additional CPU is added<br>● 100 means full load |

4. **Documentation**

   Update chaos experiment info on Spark website and developer doc.

5. **Some script**

   To help to execute chaos experiments in apache spark.

**Appendix C. Optional Rejected Designs: What alternatives were considered? Why were they rejected? If no alternatives have been considered, the problem needs more thought.**

Optional design is to use chaos-mesh-action in GA, like this PR. I finally reject this design because chaos experiment execution in the local environment could help developers to explore more production environmental issues more swiftly.