Chaos Mesh®

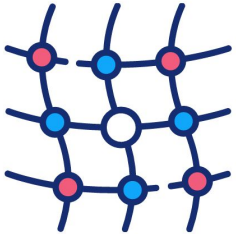
Test Applications' Storage Stability by Injecting Storage Errors

Keao Yang, Maintainer of Chaos Mesh and R&D engineer at PingCAP

About Chaos Mesh & IOChaos

- Chaos Mesh is a Chaos Engineering Platform for Kubernetes
- You can simulate a file system error by submitting a YAML file :)

```yaml
apiVersion: chaos-mesh.org/v1alpha1
kind: IoChaos
metadata:
  name: io-delay-example
  namespace: chaos-testing
spec:
  action: latency
  mode: one
  selector:
    labelSelectors:
      app: etcd
  volumePath: /var/run/etcd
  path: /var/run/etcd/**/*
  delay: "10ms"
  percent: 10
  duration: "400s"
  scheduler:
    cron: "@every 10m"
```



Chaos Mesh®

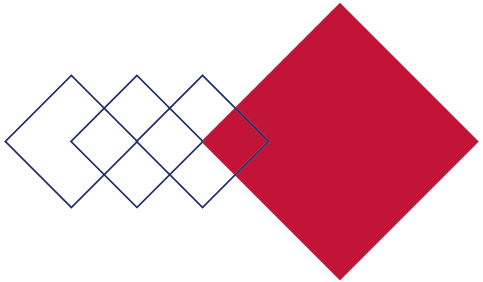
A Powerful Chaos Engineering Platform for Kubernetes

Agenda

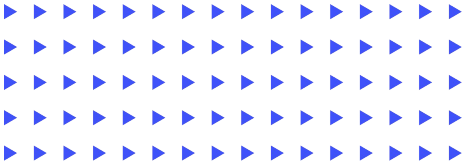
- **The storage error does matter**
  - The frequency of storage failure is really high in big cluster.
  - The impact of storage error could be amazing.

- **The implementation of storage error simulation**
  - Error injection file system
  - Replace opened resources on original file system

Chaos Mesh®



The Storage Error Does Matter



The frequency of Storage Error

Backblaze Hard Drive Failure Rates for 2020

Reporting period 1/1/2020 - 12/31/2020 inclusive

MFG	Model	Drive Size	Drive Count	Avg Age (months)	Drive Days	Drive Failures	AFR
HGST	HMS5C4040ALE640	4TB	3,100	56.65	1,083,774	8	0.27%
HGST	HMS5C4040BLE640	4TB	12,744	50.43	4,663,049	34	0.27%
HGST	HUH728080ALE600	8TB	1,075	34.85	372,000	3	0.29%
HGST	HUH721212ALE600	12TB	2,600	15.04	820,272	7	0.31%
HGST	HUH721212ALE604	12TB	2,506	3.78	275,779	9	1.19%
HGST	HUH721212ALN604	12TB	10,830	21.01	3,968,475	50	0.46%
Seagate	ST4000DM000	4TB	18,939	62.35	6,983,470	269	1.41%
Seagate	ST6000DX000	6TB	886	68.84	324,275	2	0.23%
Seagate	ST8000DM002	8TB	9,772	51.07	3,584,788	91	0.93%
Seagate	ST8000NM0055	8TB	14,406	41.34	5,286,790	177	1.22%
Seagate	ST10000NM0086	10TB	1,201	38.73	439,247	16	1.33%
Seagate	ST12000NM0007	12TB	23,036	29.78	11,947,303	339	1.04%
Seagate	ST12000NM0008	12TB	19,287	9.76	5,329,149	148	1.01%
Seagate	ST12000NM001G	12TB	7,130	6.08	1,296,149	30	0.84%
Seagate	ST14000NM001G	14TB	5,987	2.89	454,090	13	1.04%
Seagate	ST14000NM0138	14TB	360	1.56	5,784	0	0.00%
Seagate	ST16000NM001G	16TB	59	12.93	21,323	1	1.71%
Seagate	ST18000NM000J	18TB	60	3.27	5,820	2	12.54%
Toshiba	MD04ABA400V	4TB	99	67.29	36,234	0	0.00%
Toshiba	MG07ACA14TA	14TB	21,046	7.65	4,103,823	102	0.91%
Toshiba	MG07ACA14TEY	14TB	160	1.22	2,562	0	0.00%
Toshiba	MG08ACA16TEY	16TB	1,014	2.14	33,774	0	0.00%
WDC	WUH721414ALE6L4	14TB	6,002	1.68	229,861	1	0.16%
		Totals	**162,299**		**51,267,791**	**1,302**	**0.93%**

BACKBLAZE

Totals 0.93%

40000 disks result in 371 failure,
1 per day on average

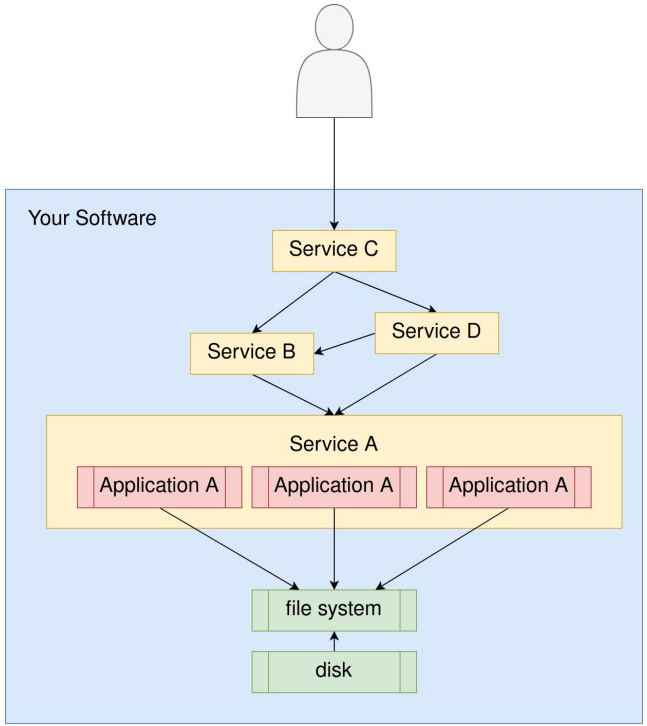
The frequency of Storage Error

Drive characteristics						Usage characteristics				Reliability metrics		
Manu-fact./ Model	Cap. (GB)	Inter-face	Flash Tech.	Lith.	PE Cycl.	OP	First Deploy-ment	Drive Power Years	Rated Life Used (%)	% of Sp. Blocks Cons.	Number of Bad Sectors	ARR (%) (all)
I - A	200	SAS	eMLC	2xnm	10K	28%	Apr '14	3.95	1.26 / 0	1.36 / 1	0.58 / 0	0.19
	400						Apr '14	3.93	0.52 / 0	1.54 / 1	2.96 / 0	0.16
	800						Mar '14	3.19	0.07 / 0	1.22 / 1	3.39 / 0	0.15
	1600						Mar '14	3.74	0.01 / 0	1.46 / 1	6.2 / 0	0.21
I - B	400	SAS	eMLC	1xnm	10K	28%	Dec '15	2.69	3.99 / 3	2.49 / 2	0.9 / 0	0.14
	800						Jan '16	2.58	1.87 / 2	3.7 / 4	4.62 / 0	0.17
	1600						Jan '16	2.55	1.68 / 2	3.55 / 3	6.01 / 0	0.23
I - C	400	SAS	eMLC	1xnm	10K	28%	Jan '17	1.7	7.48 / 7	1.83 / 2	1.86 / 0	0.12
	800						Jan '17	1.45	6.01 / 6	4.21 / 4	5.58 / 0	0.27
	1600						Mar '17	1.53	5.62 / 5	4.19 / 4	6.41 / 0	0.09
	3800						Jan '17	1.12	5.15 / 4	9.86 / 10	14.97 / 0	0.59
I - D	3800	SAS	eMLC	1xnm	10K	28%	Jul '17	1.01	4.01 / 3	10.01 / 10	12.34 / 0	0.23
II - A	3840	SAS	3D-TLC	V2	10K	7%	Dec '15	2.57	0.09 / 0	11.88 / 12	0.03 / 0	0.31
II - B	3800	SAS	3D-TLC	V2	10K	7%	Oct '16	1.77	0.01 / 0	12.01 / 12	0.36 / 0	0.13
II - C	8000	SAS	3D-TLC	V3	10K	7%	Sep '17	1.06	0.01 / 0	12.38 / 12	0.03 / 0	0.69
	15300						Sep '16	1.21	0.06 / 0	13.13 / 13	1.35 / 0	1.13
II - D	960	SAS	3D-TLC	V3	10K	7%	Oct '16	1.54	0.06 / 0	16.05 / 15	0.03 / 0	0.12
	3800						Oct '16	1.8	0.01 / 0	12	0.34 / 0	0.11
II - E	400	SAS	3D-TLC	V3	10K	28%	Dec '16	2.12	0.59 / 0	19.19 / 15	0.01 / 0	0.09
	800					28%	Jan '17	1.7	0.08 / 0	15.33 / 15	0 / 0	0.10
	3800					7%	Dec '16	2.02	0.05 / 0	12 / 12	0.26 / 0	0.13
II - F	400	SAS	3D-TLC	V2	10K	28%	Jan '16	2.52	0.86 / 0	12.31 / 12	0.01 / 0	0.48
	800						Feb '16	2.55	0.19 / 0	12.19 / 12	0.01 / 0	0.36
	1600						Jan '16	2.87	0.09 / 0	11.66 / 12	0.15 / 0	0.52
II - G	800	SAS	3D-TLC	V2	10K	28%	Apr '18	0.38	0.03 / 0	0 / 0	0 / 0	0.18
	960						Jan '18	0.5	0.11 / 0	0 / 0	0.03 / 0	0.18
	3800						Jan '18	2.89	0.09 / 0	11.64 / 12	0.15 / 0	0.28
	8000						May '18	0.45	0 / 0	0 / 0	0.25 / 0	0.37
	15000						May '18	0.46	0 / 0	0 / 0	0 / 0	0.53
II - H	800	SAS	cMLC	1xnm	10K	28%	Nov '14	3.61	1.34 / 0	7.49 / 7	1.36 / 0	0.10
II - I	200	SAS	eMLC	2xnm	30K	28%	Aug '09	4.83	5.24 / 2	6.7 / 6	0.31 / 0	0.07
	400						Dec '10	3.86	2.2 / 0	8.09 / 8	0.29 / 0	0.07
	800						Dec '10	4.66	0.69 / 0	6.94 / 7	4.53 / 0	0.11
II - J	400	SAS	eMLC	1xnm	10K	28%	May '15	3.37	2.32 / 0	6.9 / 7	0.08 / 0	0.18
	800						Jul '15	3.21	0.41 / 0	6.77 / 7	0.36 / 0	0.21
	1600						Jun '15	3.36	0.13 / 0	8.59 / 9	0.49 / 0	0.38
II - K	100	SATA	SLC	4xnm	100K	28%	Apr '12	0.43	1.4 / 1	3.78 / 4	3.62 / 0	0.14
II - L	100	SATA	SLC	3xnm	100K	28%	May '10	5.97	2.05 / 1	3.73 / 4	0.51 / 0	0.06
II - M	100	SATA	SLC	5xnm	100K	28%	Jul '10	4.23	1.63 / 1	14.82 / 13	0.18 / 0	0.11
III - A	200	SAS	eMLC	2xnm	30K	28%	Dec '12	5.89	3.08 / 1	0.02 / 0	6.18 / 0	0.07

Totals 0.22%

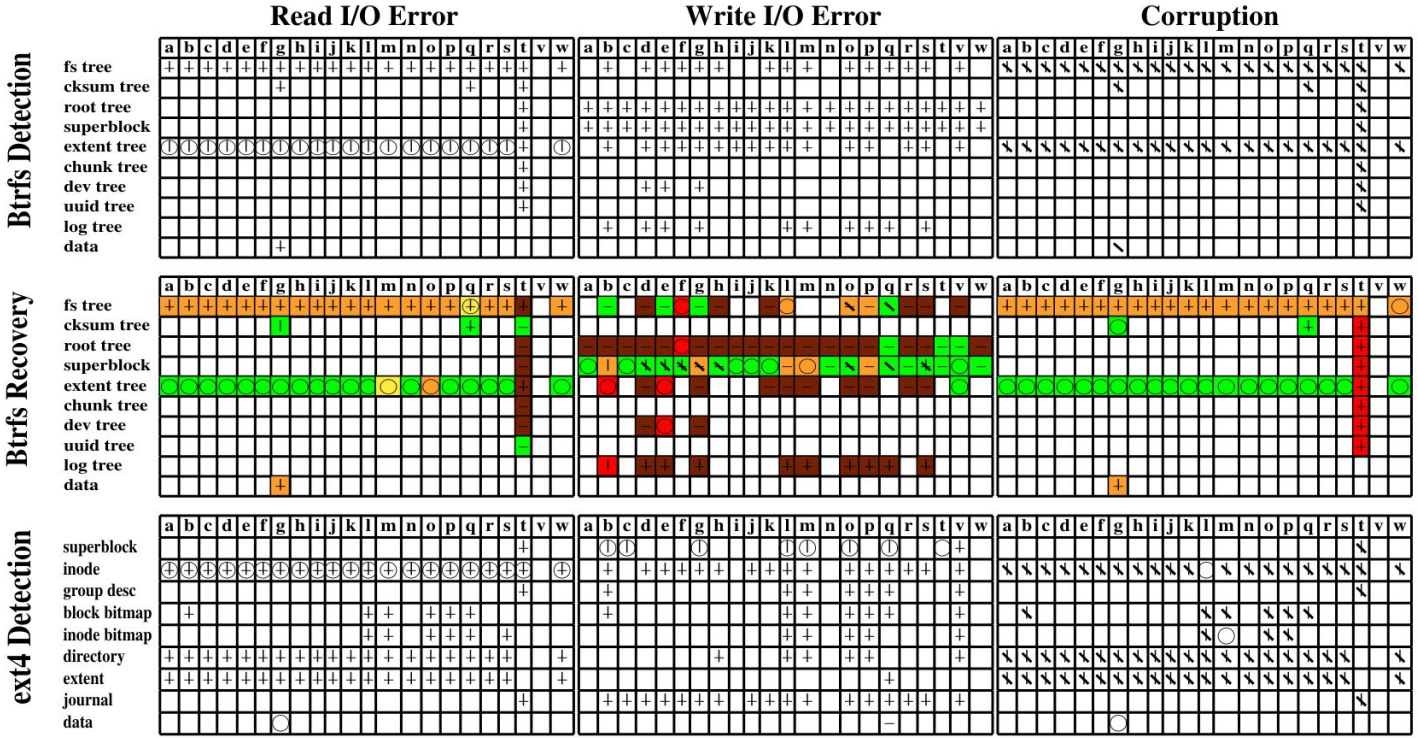
40000 disks result in 88 failure,
1 per 4 days on average

The impact of Storage Error

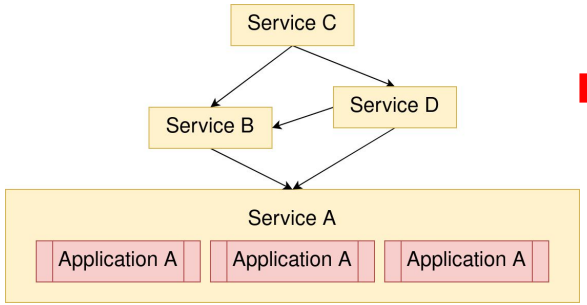


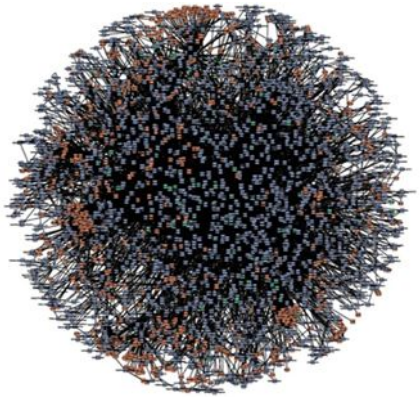
- Disk Error
- File System Corruption
- Internal Service Error
- Service Down

Reality is Even More Complicated



The Reality is Even More Complicate

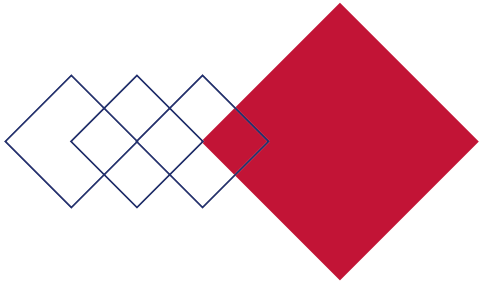




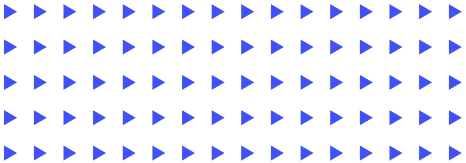
amazon.com

NETFLIX

Chaos Mesh®



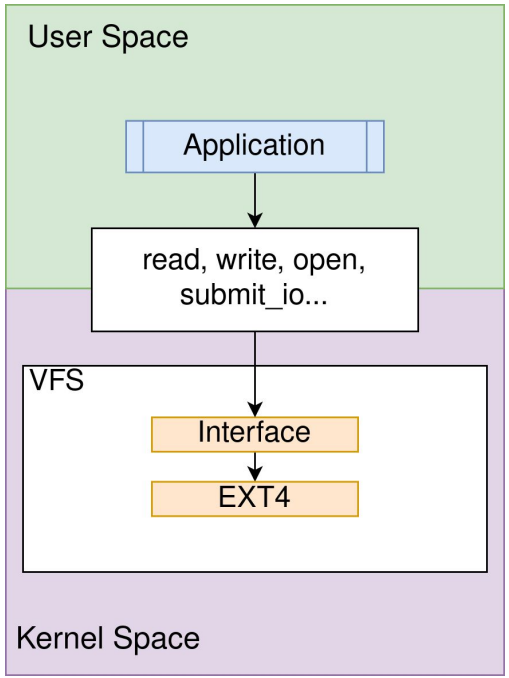
Implementation



Universal Injection Roadmap

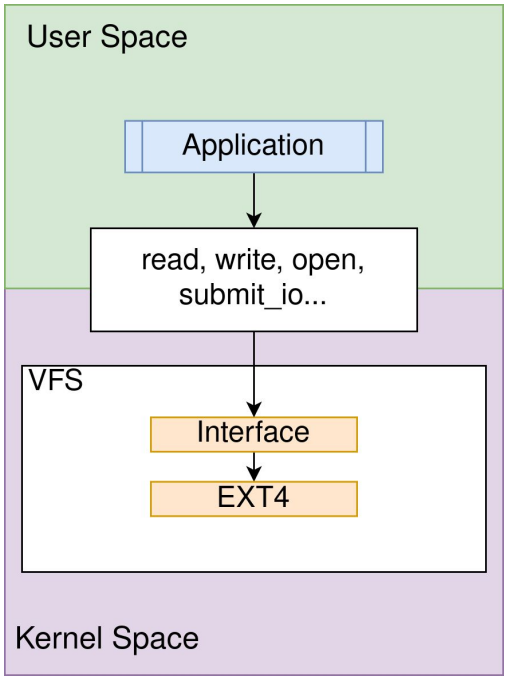
1. Investigate the normal execution path.
2. Find which step we can modify / customize to inject.

Universal Injection Roadmap



- syscall
- Page Cache
- VFS
- Real File System
- IO Scheduler
- Block Device
- Physical Device

Universal Injection Roadmap

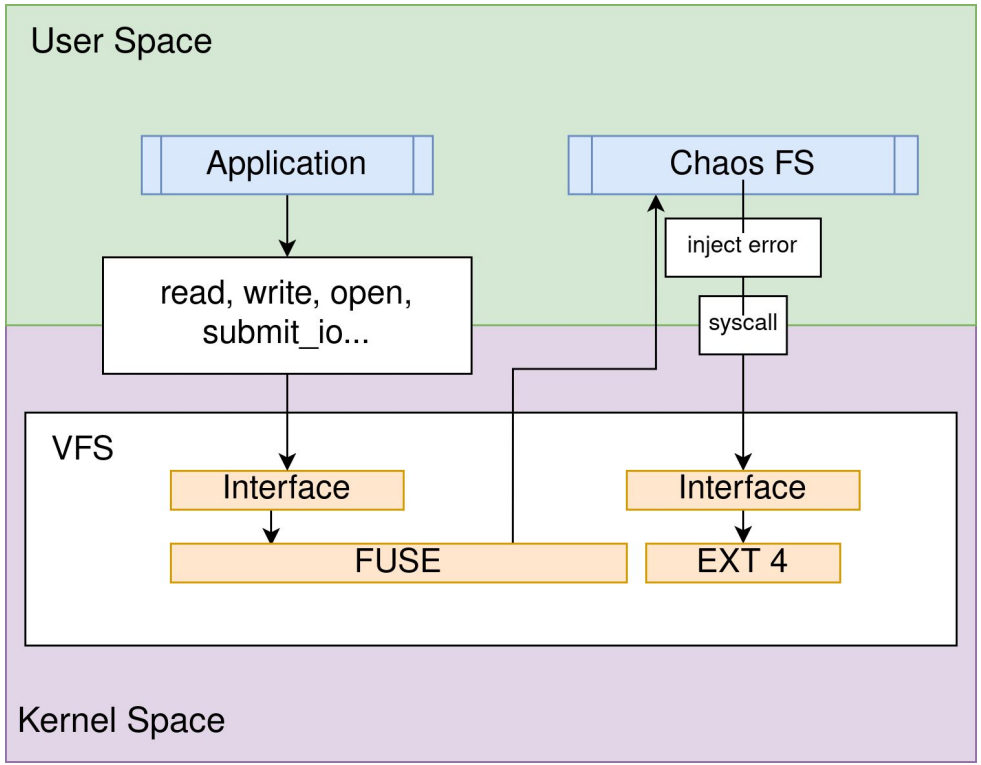


:) Better Effect

:(Higher Cost

- syscall (inject with BPF, ptrace)
- File System (inject with a customized FS)
- Block Device (provide a customized driver)
- Hard Drive (customizable device controller)

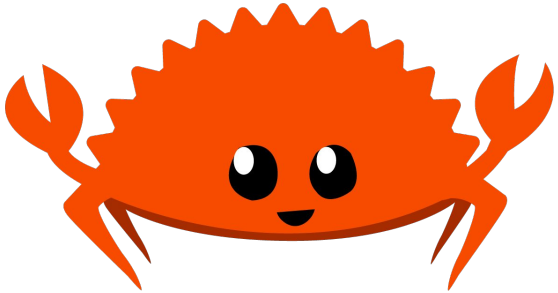
Error Injection File System



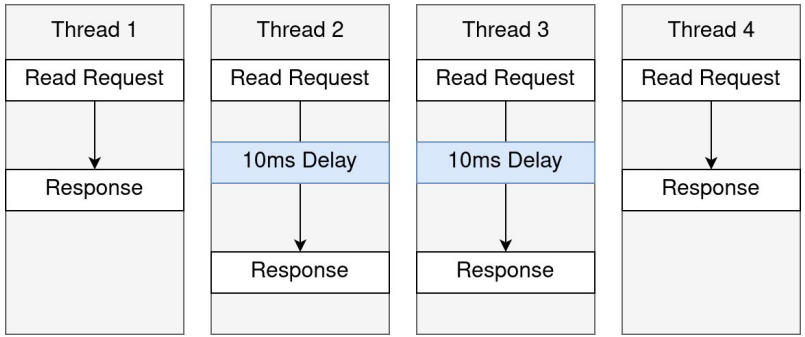
- read("/mnt/data1/hello.txt")
- "/mnt/data1" is mounted with a Chaos FS
- Chaos FS reads from *"/mnt/data1__chaosfs__/hello.txt"*
- inject delay or error
- return to FUSE

ChaosFS Implementation

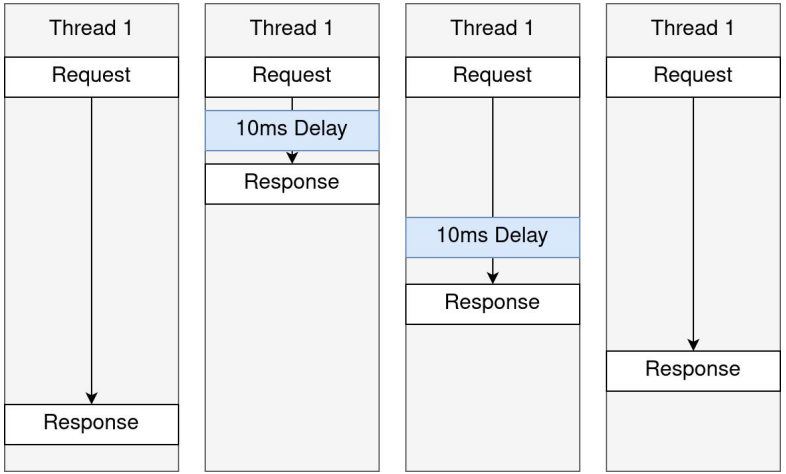
- Safe
  - No panic, no corruption
  - Data shouldn’t be lost
- Performance
  - Mutlithread
  - Asynchronous
- Rich error injection
  - Delay
  - Data corruption
  - File attributes error



Multithread & Asynchronous



:) 50% Request with 10ms delay, 5ms in AVG

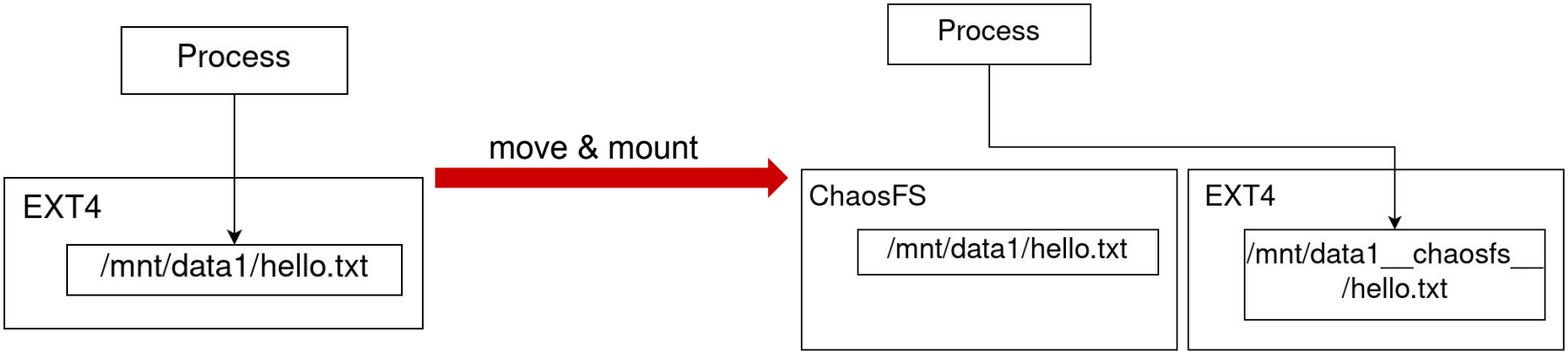


:(50% Request with 10ms delay, 10ms in AVG

Move & Mount

- mount --move /mnt/data1 /mnt/data1__chaosfs__
- mount -t chaosfs /mnt/data1

Replace Opened Resource



Replace Opened Resource

open(“/mnt/data1/hello.txt”, O_RDONLY | O_CREAT)

FD	Path
4	/mnt/data1/hello.txt

Replace Opened Resource

open(“/mnt/data1/hello.txt”, O_RDONLY | O_CREAT)

mount --move /mnt/data1 /mnt/data1__chaosfs__

FD	Path
4	/mnt/data1__chaosfs__/hello.txt

Replace Opened Resource

int dup2(int *oldfd*, **int** *newfd*);

The **dup2**() system call creates a copy of the file descriptor *oldfd*, using the the file descriptor number specified in *newfd*.

Replace Opened Resource

open(“/mnt/data1/hello.txt”, O_RDONLY | O_CREAT)

mount --move /mnt/data1 /mnt/data1__chaosfs__

open(“/mnt/data1/hello.txt”, O_RDONLY | O_CREAT)

FD	Path
4	/mnt/data1__chaosfs__/hello.txt
5	/mnt/data1/hello.txt

Replace Opened Resource

open(“/mnt/data1/hello.txt”, O_RDONLY | O_CREAT)

mount --move /mnt/data1 /mnt/data1__chaosfs__

open(“/mnt/data1/hello.txt”, O_RDONLY | O_CREAT)

dup2(5, 4)

FD	Path
4	/mnt/data1/hello.txt
5	/mnt/data1/hello.txt

Replace Opened Resource

```
open("/mnt/data1/hello.txt", O_RDONLY | O_CREAT)

mount --move /mnt/data1 /mnt/data1__chaosfs__

open("/mnt/data1/hello.txt", O_RDONLY | O_CREAT)

dup2(5, 4)
```

FD	Path
4	/mnt/data1/hello.txt
5	/mnt/data1/hello.txt

Replace Opened Resource

open(“/mnt/data1/hello.txt”, O_RDONLY | O_CREAT)

dup2(5, 4)

Replace Opened Resource

open(“/mnt/data1/hello.txt”, O_RDONLY | O_CREAT)

dup2(5, 4)

```
// open
; mov rax, 0x2
; lea rdi, [-> new_paths]
; add rdi, QWORD [r14+r15+8] // path
; mov rdx, 0x0
; syscall
; mov r12, rax // store newly opened fd in r12
// dup2
; mov rax, 0x21
; mov rdi, r12
; mov rsi, QWORD [r14+r15] // fd
; syscall
// close
; mov rax, 0x3
; mov rdi, r12
; syscall
```

Replace Opened Resource

open(“/mnt/data1/hello.txt”, O_RDONLY | O_CREAT)

dup2(5, 4)

```
// open
; mov rax, 0x2
; lea rdi, [-> new_paths]
; add rdi, QWORD [r14+r15+8] // path
; mov rdx, 0x0
; syscall
; mov r12, rax // store newly opened fd in r12
// dup2
; mov rax, 0x21
; mov rdi, r12
; mov rsi, QWORD [r14+r15] // fd
; syscall
// close
; mov rax, 0x3
; mov rdi, r12
; syscall
```

```
Disassembly of section .data:

0000000000000000 <.data>:
   0:   48 c7 c0 02 00 00 00    mov    rax,0x2
   7:   48 8d 3d f2 ff ff ff    lea    rdi,[rip+0xfffffffffffffff2]
   e:   4b 03 7c 3e 08          add    rdi,QWORD PTR [r14+r15*1+0x8]
  13:   48 c7 c2 00 00 00 00    mov    rdx,0x0
  1a:   0f 05                   syscall
  1c:   49 89 c4                mov    r12,rax
  1f:   48 c7 c0 21 00 00 00    mov    rax,0x21
  26:   4c 89 e7                mov    rdi,r12
  29:   4b 8b 34 3e             mov    rsi,QWORD PTR [r14+r15*1]
  2d:   0f 05                   syscall
```

Replace Opened Resource

1. process_vm_writev

   Load the binary codes into the target process memory space

1. ptrace

   Modify the registers and let IP points to the injected binary codes

1. restore

Mount Namespace Injection

ChaosFS doesn't exist in target Container/Mount Namespace

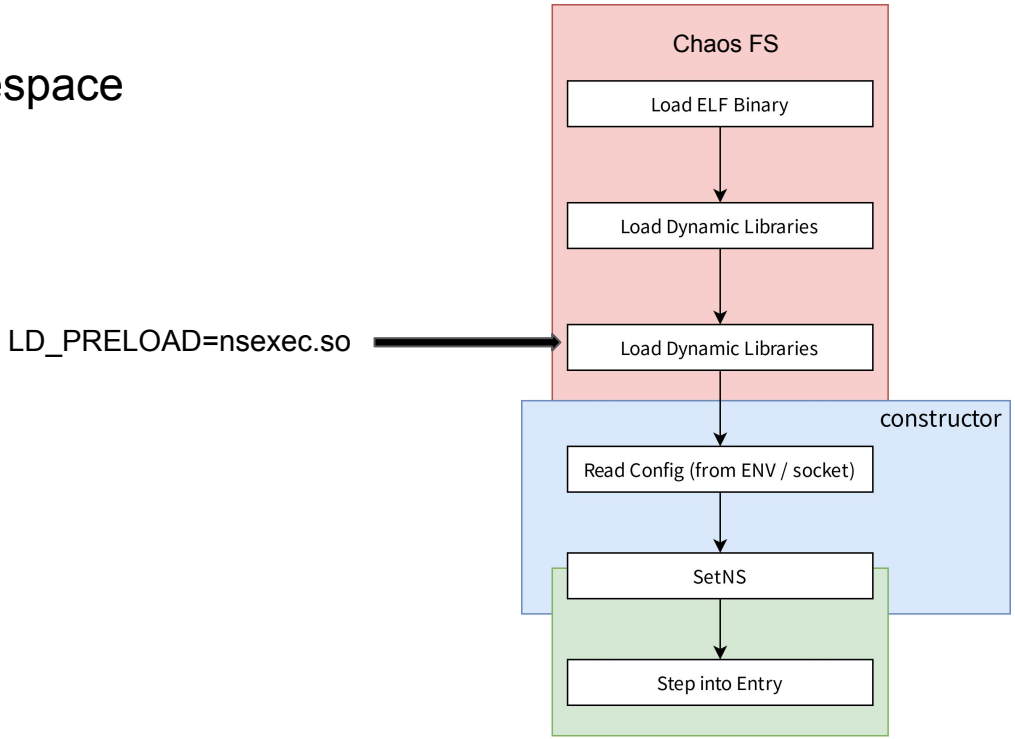
ChaosFS needs to run in target Container/Mount Namespace

Mount Namespace Injection

```
❯ sudo docker exec -it 477b5cea56e6 /bin/bash
OCI runtime exec failed: exec failed: container_linux.go:367: starting c
ontainer process caused: exec: "/bin/bash": stat /bin/bash: no such file
 or directory: unknown
```

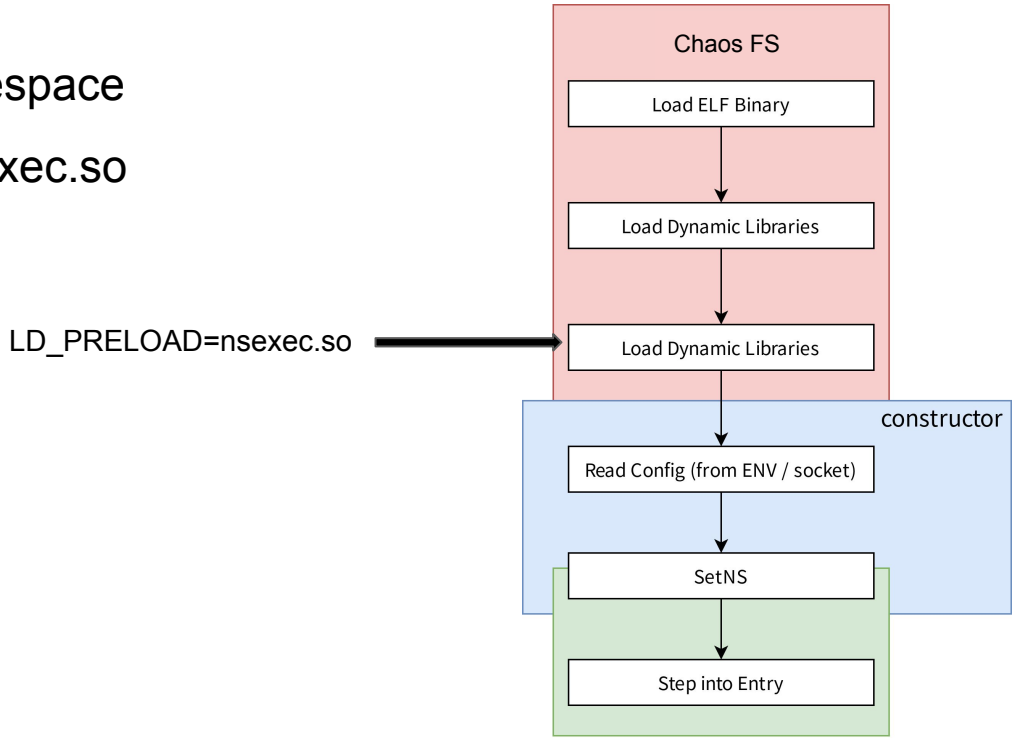
Mount Namespace Injection

1. Boot from current Mount Namespace



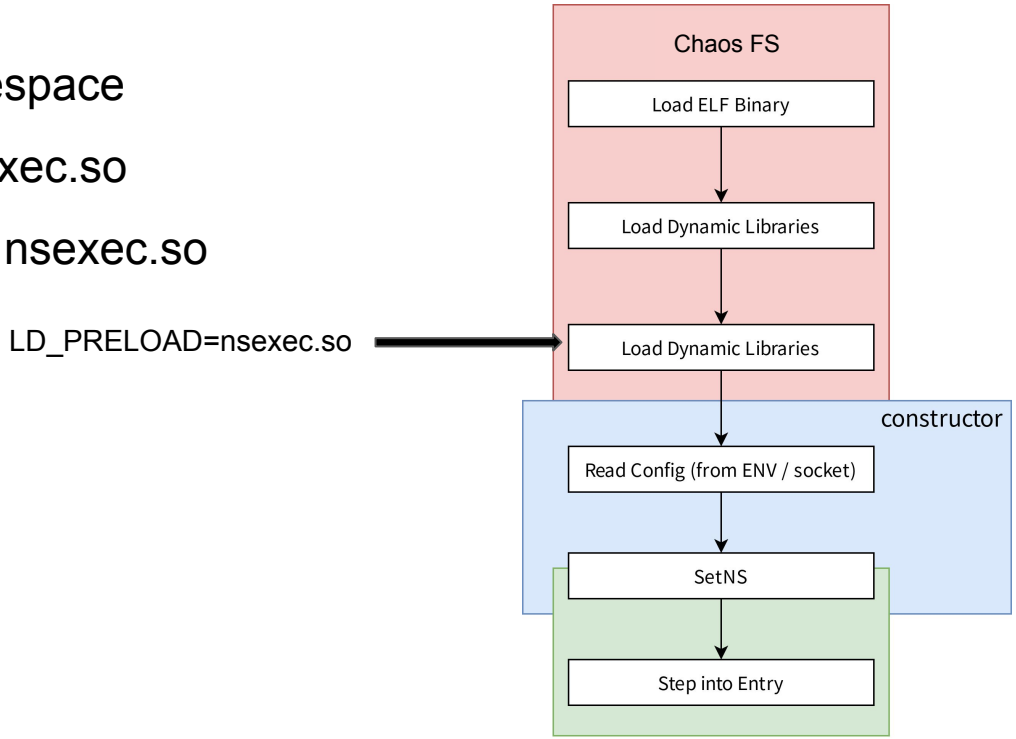
Mount Namespace Injection

1. Boot from current Mount Namespace
2. Set LD_PRELOAD to load nsexec.so



Mount Namespace Injection

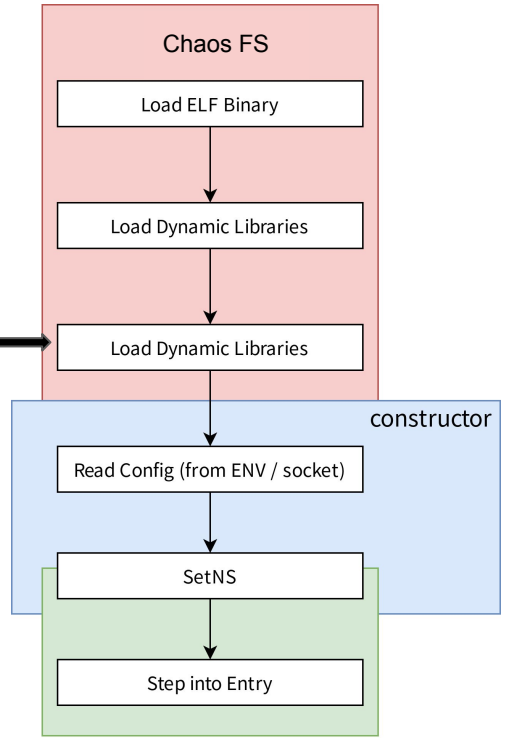
1. Boot from current Mount Namespace
2. Set LD_PRELOAD to load nsexec.so
3. Call setns in the constructor of nsexec.so



Mount Namespace Injection

1. Boot from current Mount Namespace
2. Set LD_PRELOAD to load nsexec.so
3. Call setns in the constructor of nsexec.so
4. Run in target Namespace

LD_PRELOAD=nsexec.so

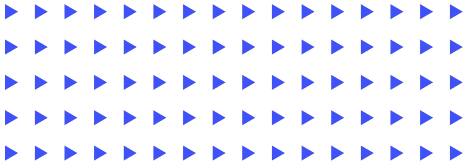


PingCAP
Make Growth Easier.

About Us

PingCAP is a software service provider committed to delivering one-stop enterprise-grade database solutions.

Chaos Mesh® is a powerful chaos engineering platform for Kubernetes.



Thank You :)

- Slack: CNCF/#project-chaos-mesh
- Official Website: chaos-mesh.org
- GitHub: chaos-mesh
- Twitter: @chaos_mesh
- Youtube: Chaos Mesh