CloudRaid: Hunting Concurrency Bugs in the Cloud via Log-Mining

Jie Lu, Feng li, Lian li, XiaoBing Feng

State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences

University of Chinese Academy of Sciences

August 14, 2019

Distributed Systems

- Open Source

Distributed Systems

- Open Source
  - YARN HDFS HBase Cassandra

Distributed Systems

- Open Source
  - YARN HDFS HBase Cassandra
- Industry
  - Aliyun AWS Azure Google cloud

Outages

AWS, February 28, 2017

- Storage Service Disruption
- Shook the industry
- About four hours
- Wrong command

Outages

Microsoft Office 365, March 21, 2017

Get in the cloud

Microsoft Office 365 brings together cloud versions of our most trusted communications and collaboration products with the latest version of our desktop suite for businesses of all sizes.

- Service become inaccessible
  - OneDrive
  - Skype
  - Outlook
- An hour
- Software bugs

Threats of Reliability

- Power Outages

[1] **gunawi2016does**.

Threats of Reliability

- Power Outages
- Security Attacks

[1] **gunawi2016does**.

Threats of Reliability

- Power Outages
- Security Attacks
- Human Errors

[1] **gunawi2016does**.

Threats of Reliability

- Power Outages
- Security Attacks
- Human Errors
- Software bugs[1]

[1] **gunawi2016does**.

Concurrency bugs

- Local

[2] **leesatapornwongsa2016taxdc**.

Concurrency bugs

- Local
  - Schedule of Threads

[2] **leesatapornwongsa2016taxdc**.

Concurrency bugs

- Local
  - Schedule of Threads
  - Well studied and detected ☺

[2] **leesatapornwongsa2016taxdc**.

Concurrency bugs

- Local
  - Schedule of Threads
  - Well studied and detected ☺
- Distributed

[2] **leesatapornwongsa2016taxdc**.

Concurrency bugs

- Local
  - Schedule of Threads
  - Well studied and detected ☺

- Distributed
  - wrong order of messages[2]

[2] **leesatapornwongsa2016taxdc**.

Concurrency bugs

- Local
  - Schedule of Threads
  - Well studied and detected ☺

figures/right.png

- Distributed
  - wrong order of messages[2]

[2] **leesatapornwongsa2016taxdc**.

Concurrency bugs

- Local
  - Schedule of Threads
  - Well studied and detected ☺

figures/error.png

- Distributed
  - wrong order of messages[2]

[2] **leesatapornwongsa2016taxdc**.

Concurrency bugs

- Local
  - Schedule of Threads
  - Well studied and detected ☺

figures/error.png

- Distributed
  - wrong order of messages[2]

[2] **leesatapornwongsa2016taxdc**.

Concurrency bugs

- Local
  - Schedule of Threads
  - Well studied and detected ☺

`figures/error.png`

- Distributed
  - wrong order of messages[2]
  - Hard to detect 😢

[2] **leesatapornwongsa2016taxdc**.

Model Checking

- Enumerating all messages order

Model Checking

- Enumerating all messages order
  - a b c

Model Checking

- Enumerating all messages order
  - a b c
  - (a,b,c) (a,c,b) (b, a, c) (b,c,a) (c, a, b) (c, b, a)

Model Checking

- Enumerating all messages order
  - a b c
  - (a,b,c) (a,c,b) **(b, a, c)** (b,c,a) (c, a, b) (c, b, a)
- Simple input of YARN: **5,495** messages

Model Checking

- Enumerating all messages order
  - a b c
  - (a,b,c) (a,c,b) (b, a, c) (b,c,a) (c, a, b) (c, b, a)
- Simple input of YARN: 5,495 messages
  - State-space explosion problem

Model Checking

- Enumerating all messages order
  - a b c
  - (a,b,c) (a,c,b) **(b, a, c)** (b,c,a) (c, a, b) (c, b, a)
- Simple input of YARN: **5,495** messages
  - State-space explosion problem
- Are all enumerations needed?

Model Checking

- Enumerating all messages order
  - a b c
  - (a,b,c) (a,c,b) (b, a, c) (b,c,a) (c, a, b) (c, b, a)
- Simple input of YARN: 5,495 messages
  - State-space explosion problem

- Are all enumerations needed?

NO.

Observations from Real Bugs

- Observation 1 :

$$message\ order \Longrightarrow handler\ order \Longrightarrow shared\ object\ access\ order$$

[3] **leesatapornwongsa2016taxdc**.

Observations from Real Bugs

- Observation 1 :

$$message\ order \Longrightarrow handler\ order \Longrightarrow shared\ object\ access\ order$$

- Observation 2 : Many message orders are already tested in live systems

[3] **leesatapornwongsa2016taxdc**.

Observations from Real Bugs

- Observation 1 :

$$\mathit{message\ order} \Longrightarrow \mathit{handler\ order} \Longrightarrow \mathit{shared\ object\ access\ order}$$

- Observation 2 : Many message orders are already tested in live systems
- Observation 3[3]: Most distributed concurrency bugs can be triggered by reordering only **two** messages

[3] **leesatapornwongsa2016taxdc**.

Suspicious message order

- Observation 3 : $\langle S, P \rangle$

Suspicious message order

- Observation 3 : $\langle S, P \rangle$
- $Observation1 \Rightarrow Rule1$: S and P access the same object

Suspicious message order

- Observation 3 : $\langle S, P \rangle$
- $Observation1 \Rightarrow Rule1$: S and P access the same object
- $Observation2 \Rightarrow Rule2$: The order should not be tested in live system

Information from live system

- Non - interference in live system!

Information from live system

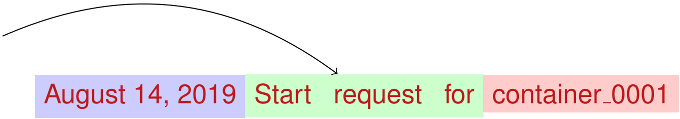
- Non - interference in live system!
- Distributed systems produce massive logs for post-mortem debug!

Information from live system

- Non - interference in live system!
- Distributed systems produce massive logs for post-mortem debug!
- Infomation

Information from live system

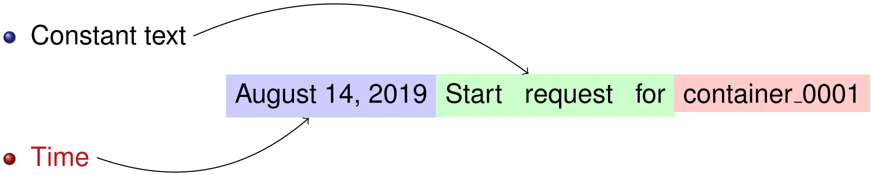
- Non - interference in live system!
- Distributed systems produce massive logs for post-mortem debug!
- Infomation
  - Constant text



August 14, 2019 Start request for container_0001

Information from live system

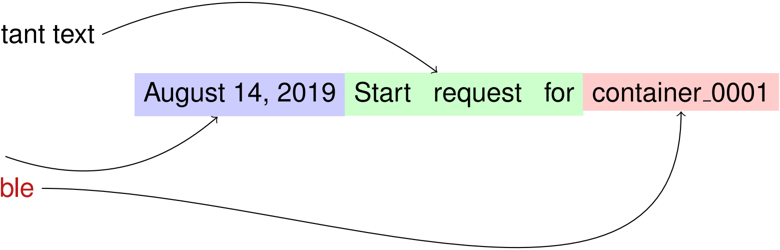
- Non - interference in live system!
- Distributed systems produce massive logs for post-mortem debug!
- Infomation
  - Constant text
  - Time



August 14, 2019 Start request for container_0001

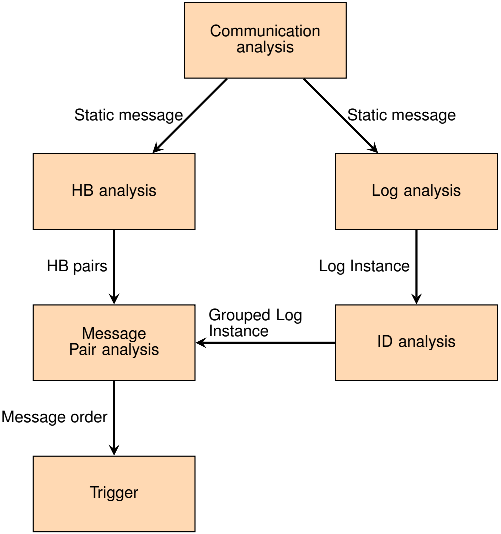
Information from live system

- Non - interference in live system!
- Distributed systems produce massive logs for post-mortem debug!
- Infomation
  - Constant text
  - Time
  - Variable



August 14, 2019 Start request for container_0001

Overveiw of CloudRaid



Communication analysis

Static message

Static message

HB analysis

Log analysis

HB pairs

Log Instance

Message Pair analysis

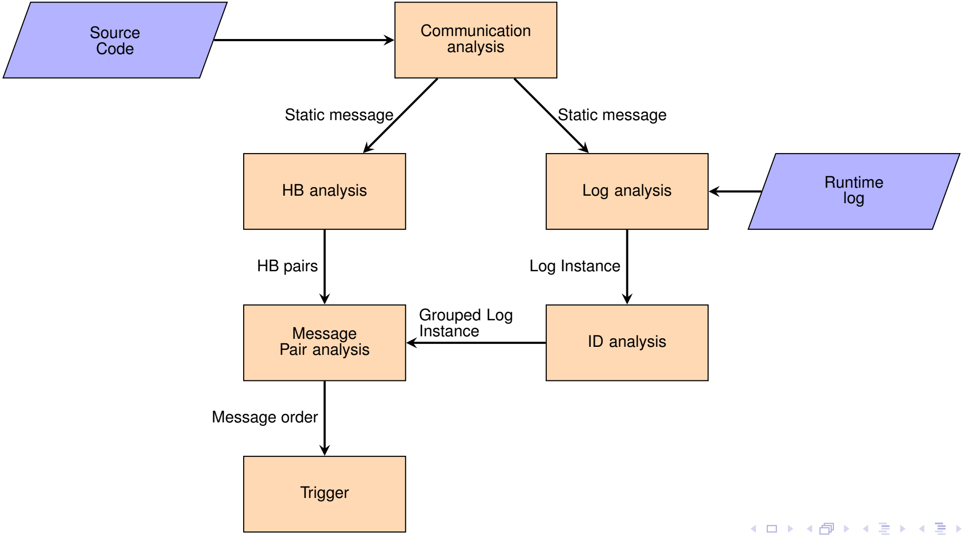
Grouped Log Instance

ID analysis

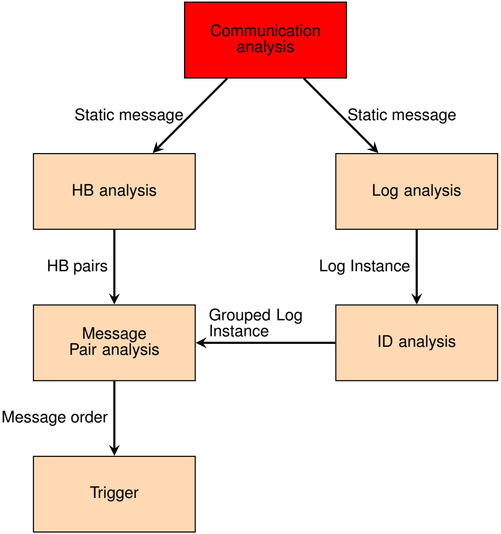
Message order

Trigger

Overveiw of CloudRaid



Overveiw of CloudRaid



Communication analysis

- Target messages?

Communication analysis

- Target messages?
    - RPC, Event, Thread

Communication analysis

- Target messages?
  - RPC, Event, Thread
  - $Client \Rightarrow Server$

```
//Client
public void EventProcessor.run() {
   ......
   response = proxy.startContainer(new StartConReq());
   ......
}
//Server
public StartConRes startContainer(StartConReq req) {

   ID containerID = req.getConLauContext().getConId();

   LOG.info("Start request for " + containerID);
   ......
}
```

Communication analysis

- Target messages?
  - RPC, Event, Thread
  - $Client \Rightarrow Server$
  - Static analysis

```
//Client
public void EventProcessor.run() {
    ......
    response = proxy.startContainer(new StartConReq());
    ......
}
//Server
public StartConRes startContainer(StartConReq req) {

    ID containerID = req.getConLauContext().getConId();

    LOG.info(”Start request for ” + containerID);
    ......
}
```

Communication analysis

- Target messages?
  - RPC, Event, Thread
  - $Client \Rightarrow Server$
  - Static analysis

- Static message:$\langle C, F, L\rangle$

```
//Client
public void EventProcessor.run() {
    ......
    response = proxy.startContainer(new StartConReq());
    ......
}
//Server
public StartConRes startContainer(StartConReq req) {

    ID containerID = req.getConLauContext().getConId();

    LOG.info(”Start request for ” + containerID);
    ......
}
```

Communication analysis

- Target messages?
  - RPC, Event, Thread
  - $Client \Rightarrow Server$
  - Static analysis
- Static message:$\langle C, F, L\rangle$
  - C: Client

```
//Client
public void EventProcessor.run() {
   ......
   response = proxy.startContainer(new StartConReq());
   ......
}
//Server
public StartConRes startContainer(StartConReq req) {

   ID containerID = req.getConLauContext().getConId();

    LOG.info(”Start request for ” + containerID);
   ......
}
```

Communication analysis

- Target messages?
  - RPC, Event, Thread
  - $Client \Rightarrow Server$
  - Static analysis
- Static message:$\langle C, F, L\rangle$
  - C: Client
  - F: Server

```
//Client
public void EventProcessor.run() {
    ......
    response = proxy.startContainer(new StartConReq());
    ......
}
//Server
public StartConRes startContainer(StartConReq req) {

    ID containerID = req.getConLauContext().getConId();

    LOG.info("Start request for " + containerID);
    ......
}
```

Communication analysis

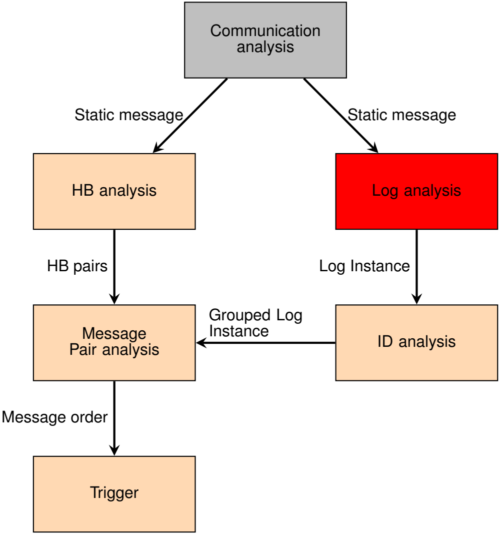
- Target messages?
  - RPC, Event, Thread
  - $Client \Rightarrow Server$
  - Static analysis

```
//Client
public void EventProcessor.run() {
    ......
    response = proxy.startContainer(new StartConReq());
    ......
}
//Server
public StartConRes startContainer(StartConReq req) {
    ID containerID = req.getConLauContext().getConId();
    LOG.info("Start request for " + containerID);
    ......
}
```

- Static message:$\langle C, F, L \rangle$
  - C: Client
  - F: Server
  - L: Log pattern

(.*) Start request for (.*)

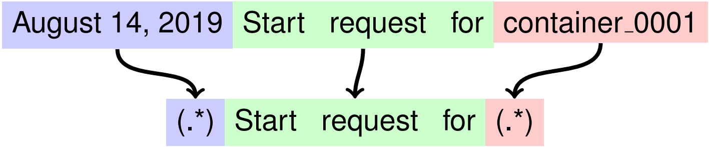
Log analysis



Log analysis

```
August 14, 2019 Start request for container_0001
```

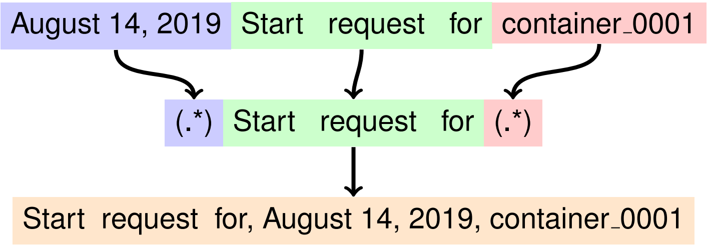
Log analysis



August 14, 2019 Start request for container_0001

(.*) Start request for (.*)

Log analysis

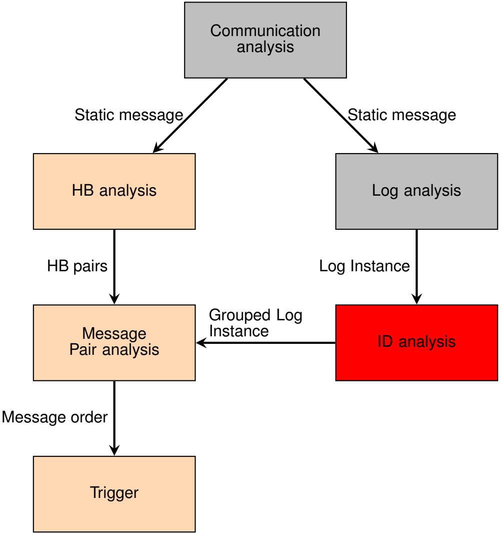


August 14, 2019 Start request for container_0001

(.*) Start request for (.*)

Start request for, August 14, 2019, container_0001

ID analysis


Communication analysis
Static message
Static message
HB analysis
Log analysis
HB pairs
Log Instance
Message Pair analysis
Grouped Log Instance
ID analysis
Message order
Trigger

ID analysis

- ID is a variable

ID analysis

- ID is a variable
- The value of one ID can index a task

ID analysis

- ID is a variable
- The value of one ID can index a task
- What variable is ID?

```
//Server
public StartConRes startContainer(StartConReq   req ) {

   ID containerID = req .getConLauContext(). getConId ();

   LOG.info(”Start request for ” +  containerID );
   ......
}
```

ID analysis

- ID is a variable
- The value of one ID can index a task
- What variable is ID?
  - propagated from the arguments of message handler

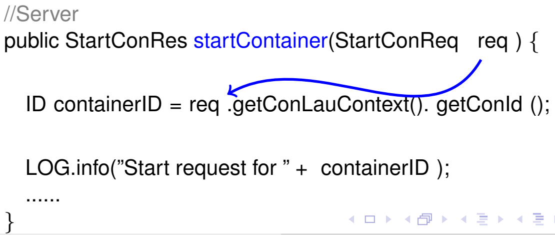
```
//Server
public StartConRes startContainer(StartConReq   req ) {

   ID containerID = req .getConLauContext(). getConId ();

   LOG.info(”Start request for ” +  containerID );
   ......
}
```

ID analysis

- ID is a variable
- The value of one ID can index a task
- What variable is ID?
  - propagated from the arguments of message handler



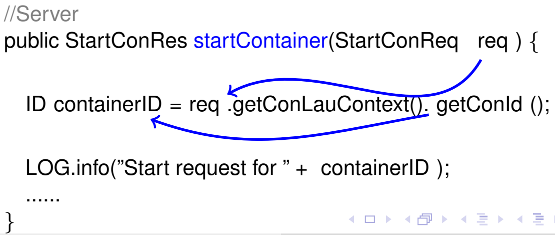
```
//Server
public StartConRes startContainer(StartConReq   req ) {

   ID containerID = req .getConLauContext(). getConId ();

   LOG.info("Start request for " +  containerID );
   ......
}
```

ID analysis

- ID is a variable
- The value of one ID can index a task
- What variable is ID?
  - propagated from the arguments of message handler



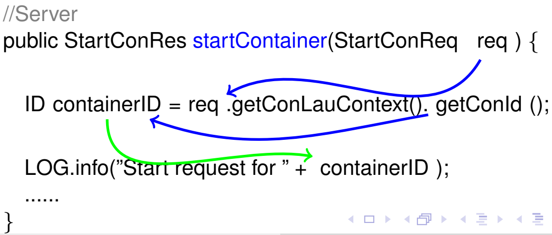
```
//Server
public StartConRes startContainer(StartConReq  req ) {

   ID containerID = req .getConLauContext(). getConId ();

   LOG.info("Start request for " +  containerID );
   ......
}
```

ID analysis

- ID is a variable
- The value of one ID can index a task
- What variable is ID?
  - propagated from the arguments of message handler
  - Printed in log

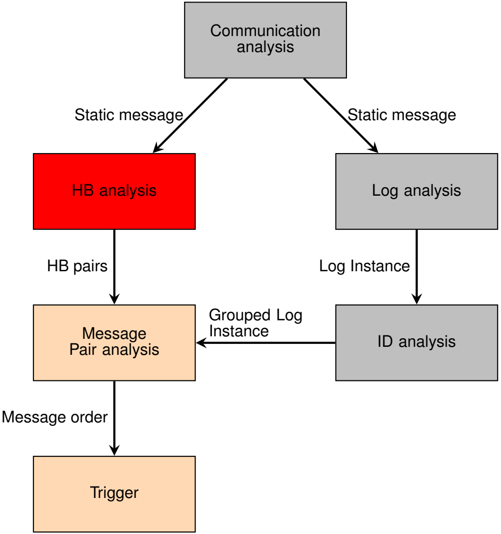


```
//Server
public StartConRes startContainer(StartConReq  req ) {

  ID containerID = req .getConLauContext(). getConId ();

  LOG.info("Start request for " +  containerID );
  ......
}
```

Happend Before analysis



Happend Before analysis

- Static analysis

Happend Before analysis

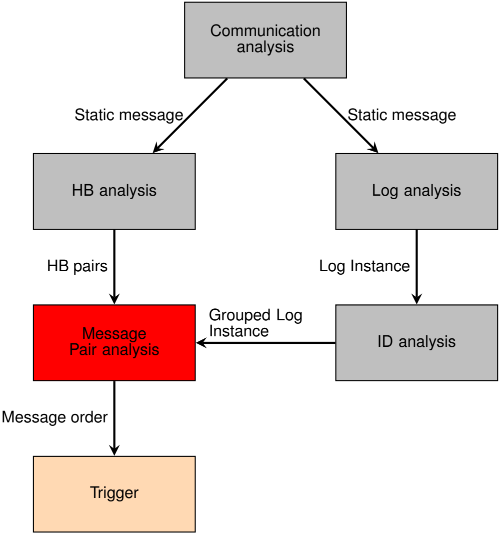
- Static analysis

$$S :< C_S, F_S, L_S >, P :< C_P, F_P, L_P >$$

$$If\ C_S\ is\ in\ F_P,\ then\ P\ HB\ S$$

- Fix-point

Message Pair analysis

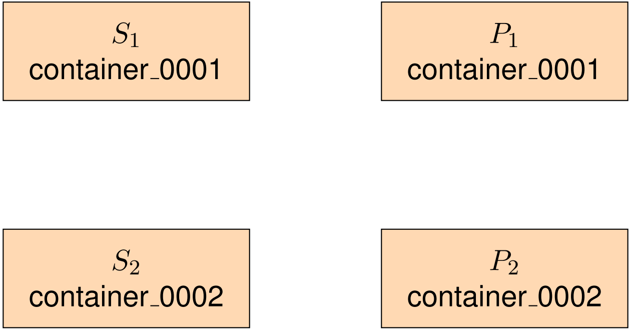


Message Pair analysis

- Dynamic messages

Message Pair analysis

- Dynamic messages
  - Static Message, RunTime logs



S_1
container_0001

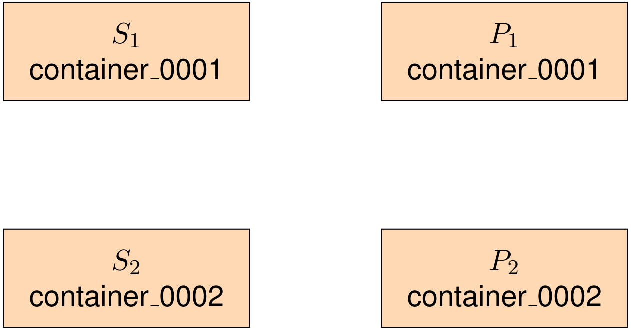
P_1
container_0001

S_2
container_0002

P_2
container_0002

Message Pair analysis

- Dynamic messages
  - Static Message, RunTime logs
  - ID value to group messages that belong to same task



S_1
container_0001

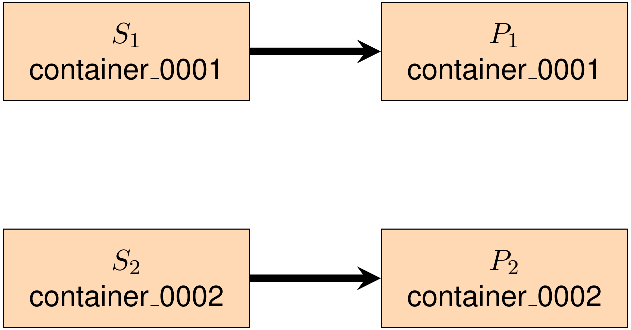
P_1
container_0001

S_2
container_0002

P_2
container_0002

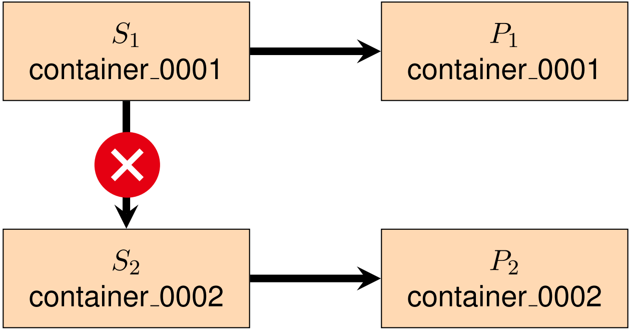
Message Pair analysis

- Dynamic messages
  - Static Message, RunTime logs
  - ID value to group messages that belong to same task
  - Time



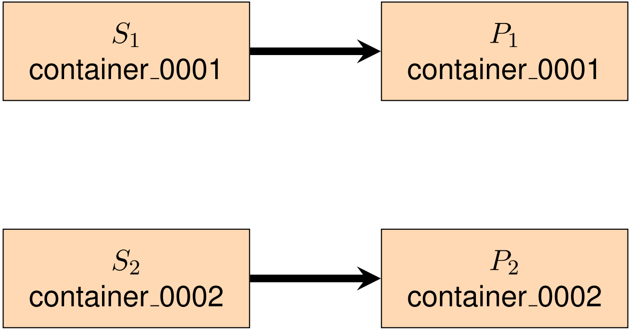
Message Pair analysis

- Dynamic messages
  - Static Message, RunTime logs
  - ID value to group messages that belong to same task
  - Time
- Same Shared Object(rule1) : ID value→same task



Message Pair analysis

- Dynamic messages
  - Static Message, RunTime logs
  - ID value to group messages that belong to same task
  - Time
- Same Shared Object(rule1) : ID value→same task
- Filter executed order pairs(rule2)



S_1
container_0001

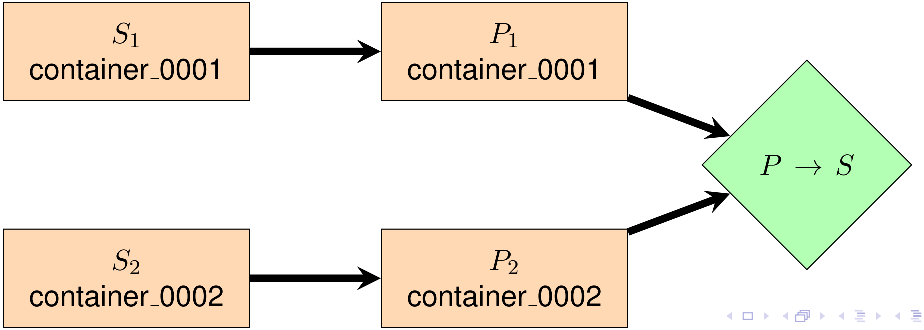
P_1
container_0001

S_2
container_0002

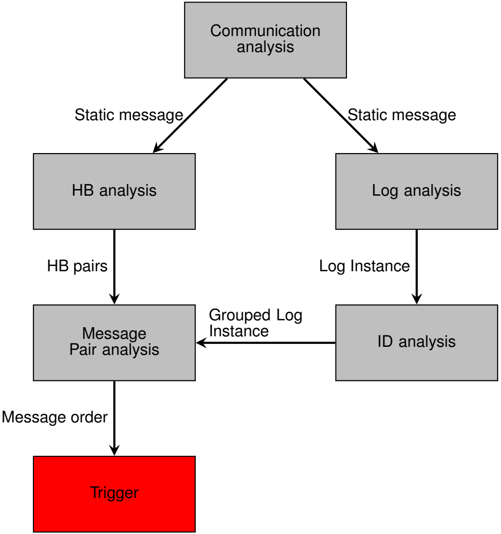
P_2
container_0002

Message Pair analysis

- Dynamic messages
  - Static Message, RunTime logs
  - ID value to group messages that belong to same task
  - Time
- Same Shared Object(rule1) : ID value→same task
- Filter executed order pairs(rule2)

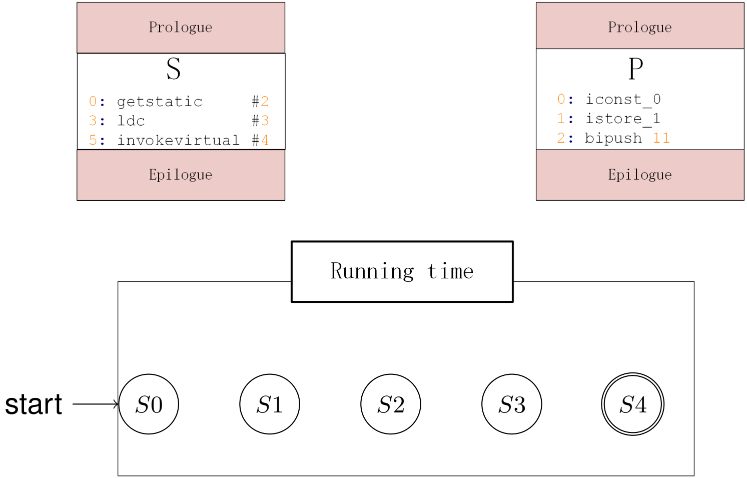


Trigger



Communication analysis
Static message
Static message
HB analysis
Log analysis
HB pairs
Log Instance
Message Pair analysis
Grouped Log Instance
ID analysis
Message order
Trigger

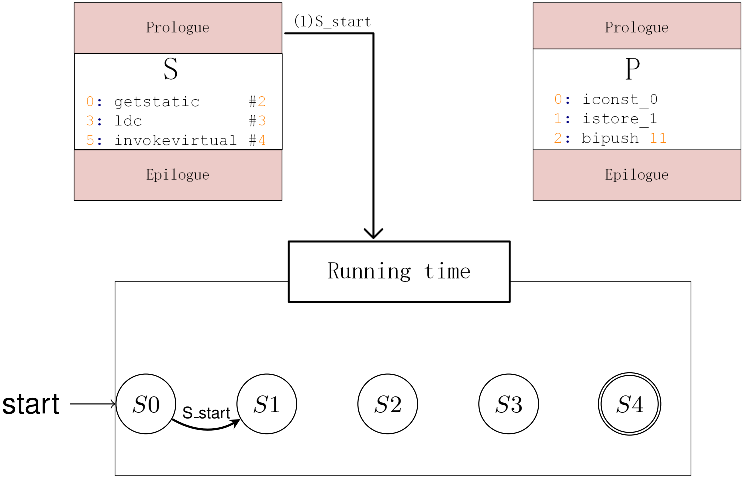
Trigger



```
0: getstatic       #2
3: ldc             #3
5: invokevirtual  #4
```

```
0: iconst_0
1: istore_1
2: bipush 11
```

Trigger



Prologue

S

```
0: getstatic      #2
3: ldc            #3
5: invokevirtual  #4
```

Epilogue

(1)S_start

Prologue

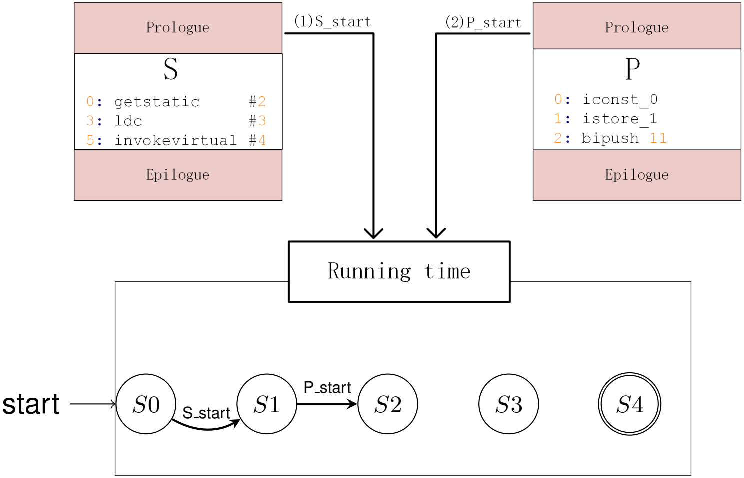
P

```
0: iconst_0
1: istore_1
2: bipush 11
```

Epilogue

Running time

Trigger



Prologue

S

```
0: getstatic       #2
3: ldc             #3
5: invokevirtual #4
```

Epilogue

(1)S_start

(2)P_start

Prologue

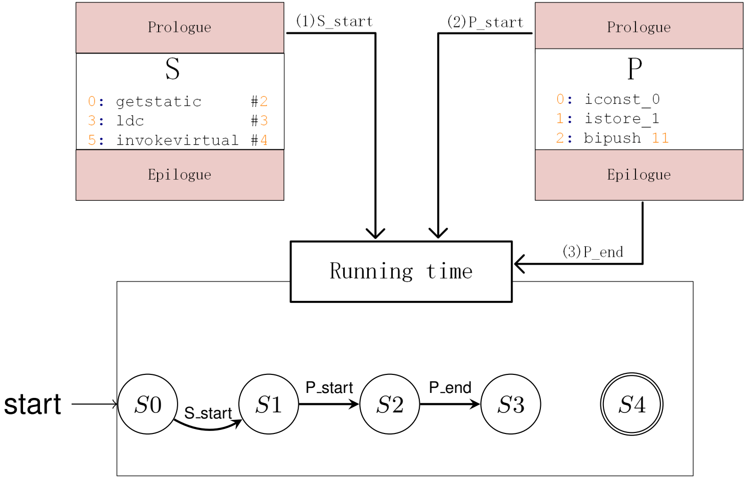
P

```
0: iconst_0
1: istore_1
2: bipush 11
```

Epilogue

Running time

Trigger



(1)S_start

(2)P_start

(3)P_end

Prologue

S

```
0: getstatic      #2
3: ldc            #3
5: invokevirtual  #4
```

Epilogue

Prologue

P

```
0: iconst_0
1: istore_1
2: bipush 11
```

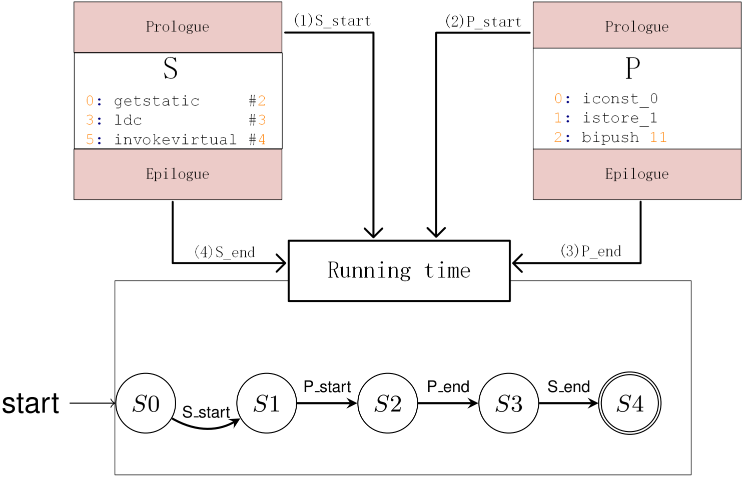
Epilogue

Running time

start

$S0$ S_start $S1$ P_start $S2$ P_end $S3$ $S4$

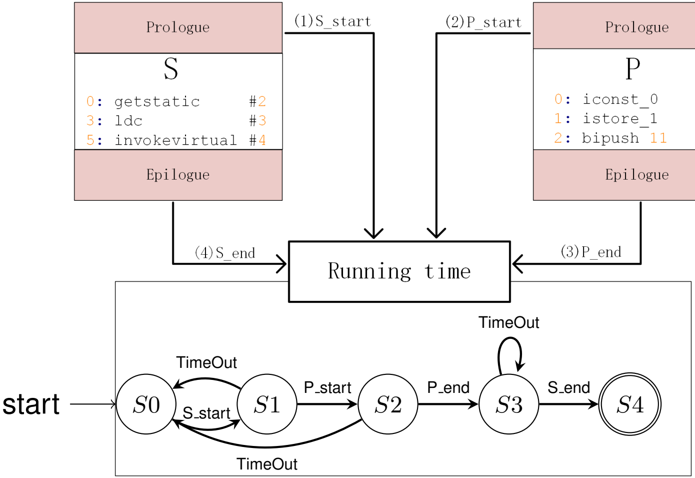
Trigger



```
0: getstatic       #2
3: ldc             #3
5: invokevirtual  #4
```

```
0: iconst_0
1: istore_1
2: bipush 11
```

Trigger



```
0: getstatic       #2
3: ldc             #3
5: invokevirtual #4
```

```
0: iconst_0
1: istore_1
2: bipush 11
```

Evaluation

Table: Systems under testing.

System	# CloudRaid code changes	Workload
Hadoop2/Yarn	48	wordcount + kill
HDFS	18	putfile + reboot
HBase	25	write + node crash
Cassandra	17	write

Evaluation

Table: Message orders pruned by each analysis.

System	#Total	% of Pruned			
		HB	Order	ID	All
Hadoop2/Yarn	4489	1.0%	11.1%	81.5%	93.6%
HDFS	81	2.5%	45.7%	51.9%	85.2%
HBase	324	2.5%	57.7%	34.3%	94.4%
Cassandra	16	0.0%	75.0%	0.0%	75%

Evaluation

Table: Analysis and testing times of CloudRaid.

System	Profiling(s)	Analysis(s)	Trigger(s)
Hadoop2/Yarn	648.0	131.3	6990.2
HDFS	646.0	60.0	828.3
HBase	1309.0	63.3	1368.0
Cassandra	263.1	112.3	60.3

Evaluation

Table: 28 bugs detected and 8 of them are new bugs. All bugs are confirmed by the original developers, and 3 of them are already fixed.

Bug ID	type	status	Patched?	Symptom
YARN-6948	Order	Fixed	yes	Attempt fail
YARN-6969	Order	Unresolved	no	Wrong state
YARN-7176	Atomicity	Unresolved	yes	Cluster down
YARN-7563	Order	Unresolved	yes	Resource leak
YARN-7663	Order	Fixed	yes	Job fail
YARN-7726	Order	Unresolved	yes	Wrong state
YARN-7786	Order	Fixed	yes	Null Pointer
HBase-19004	Order	Unresolved	no	Data loss

Case study

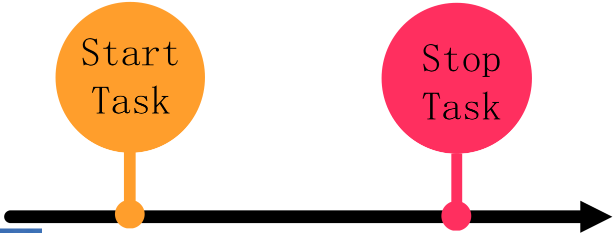
Why CloudRaid works

Case study

Why CloudRaid works

Case study

Why CloudRaid works



Distributed systems always log important messages

Case study

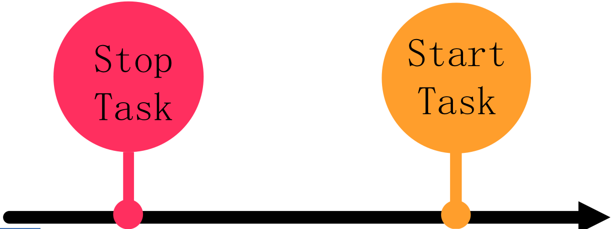
Why CloudRaid works

Stop Task

Distributed systems always log important messages

Case study

Why CloudRaid works

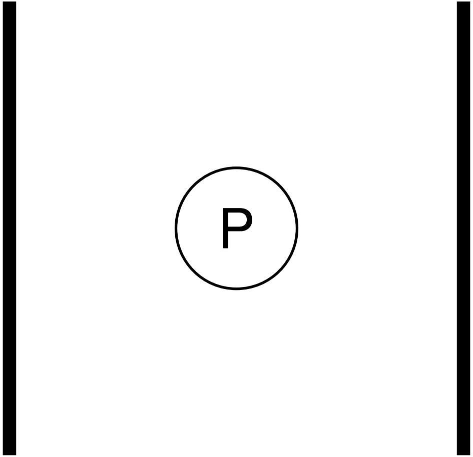


Stop Task

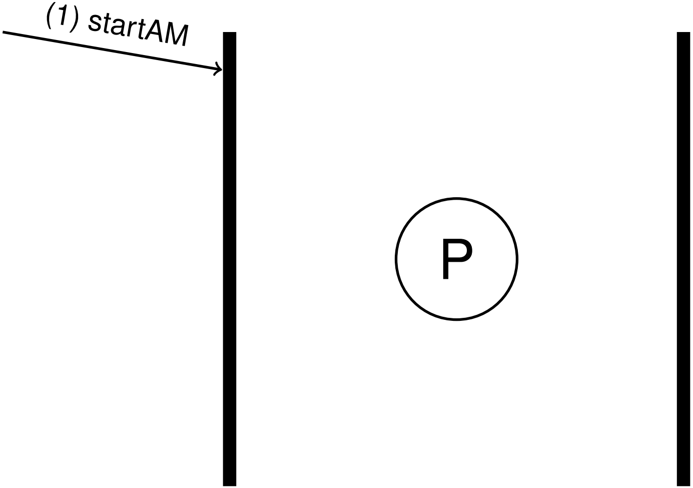
Start Task

Distributed systems always log important messages

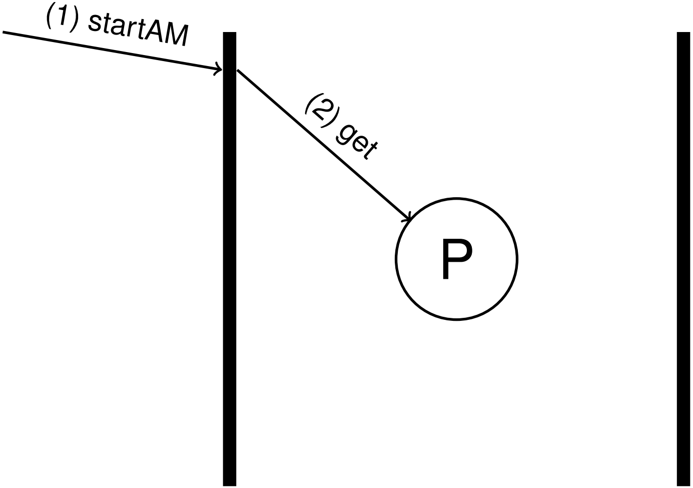
YARN-7786



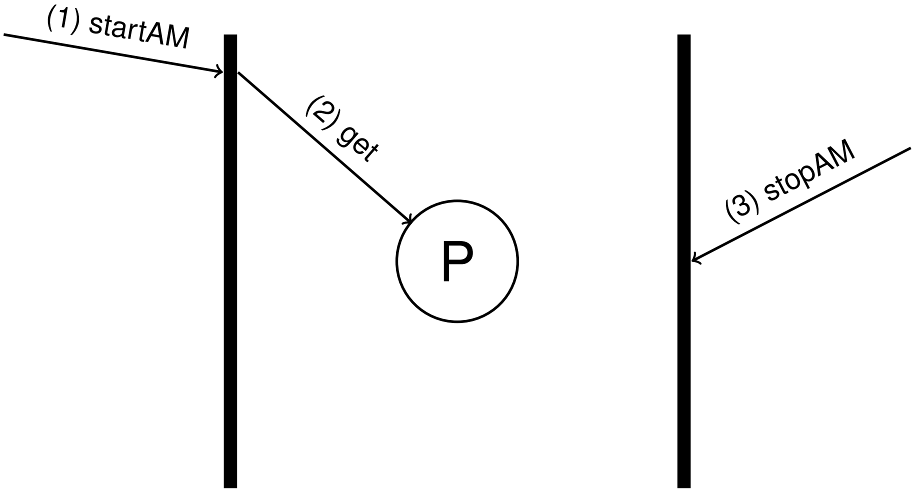
YARN-7786



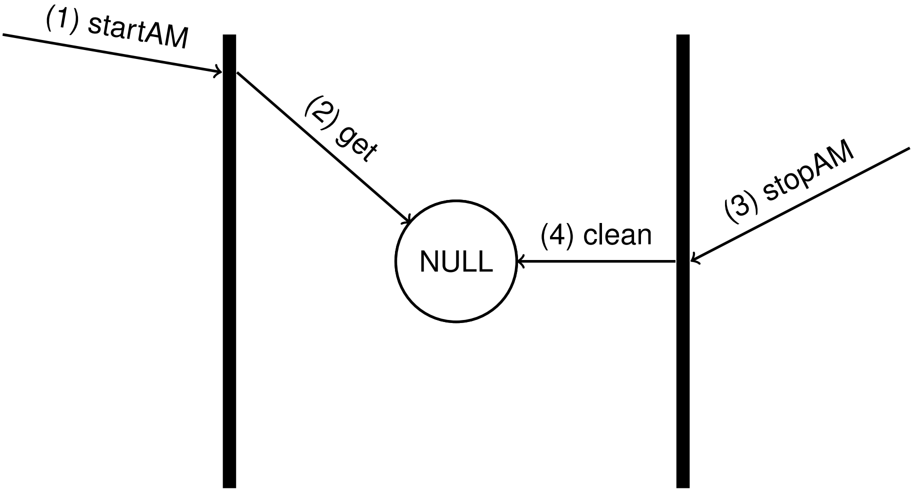
YARN-7786



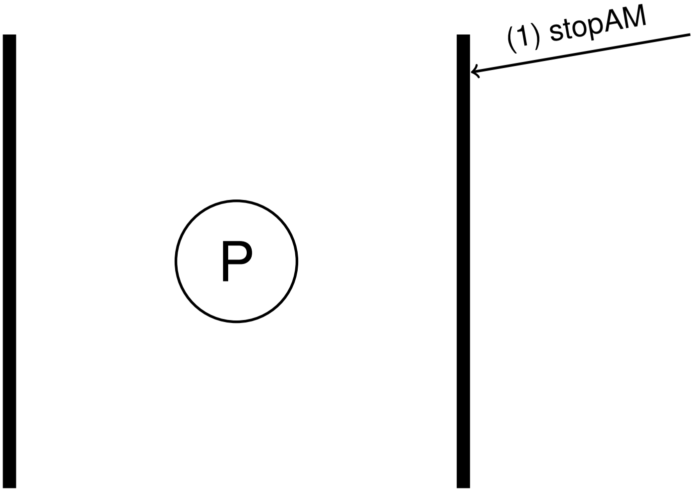
YARN-7786



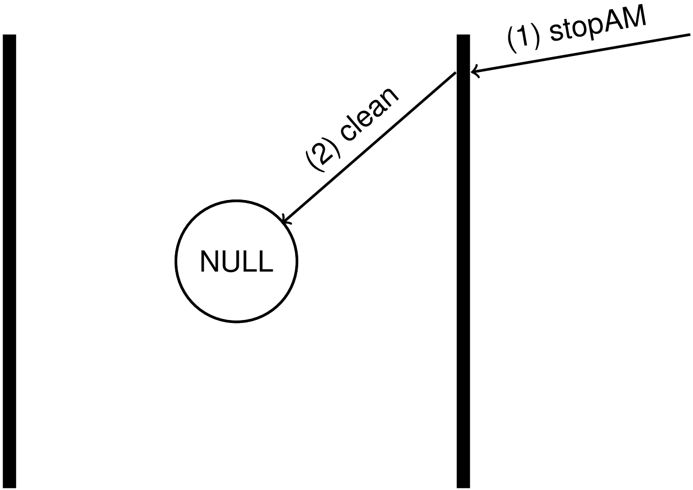
YARN-7786



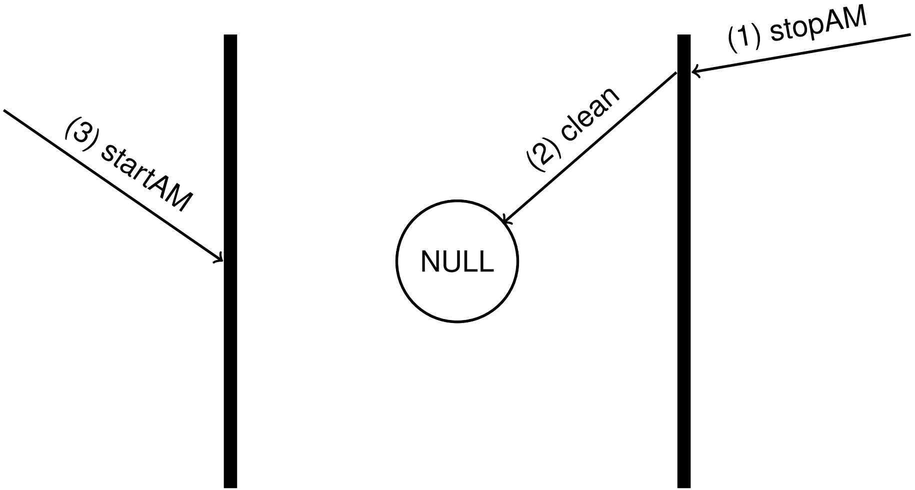
YARN-7786



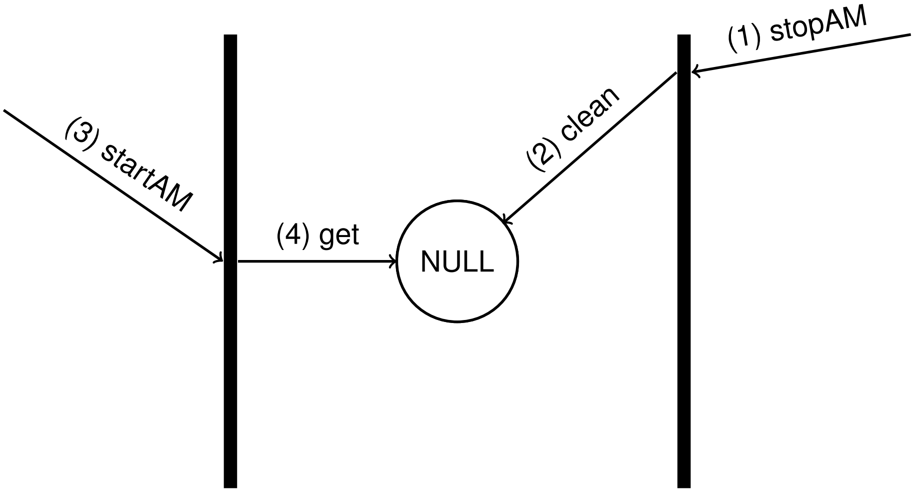
YARN-7786



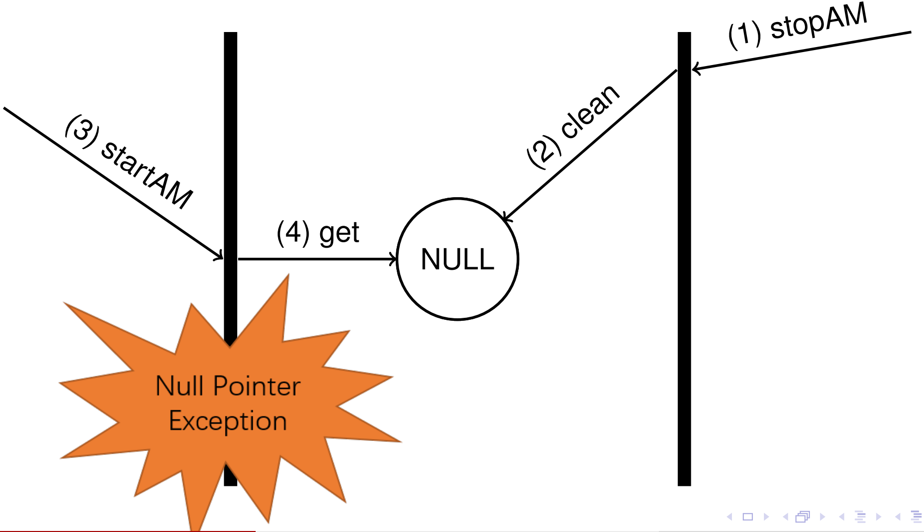
YARN-7786



YARN-7786

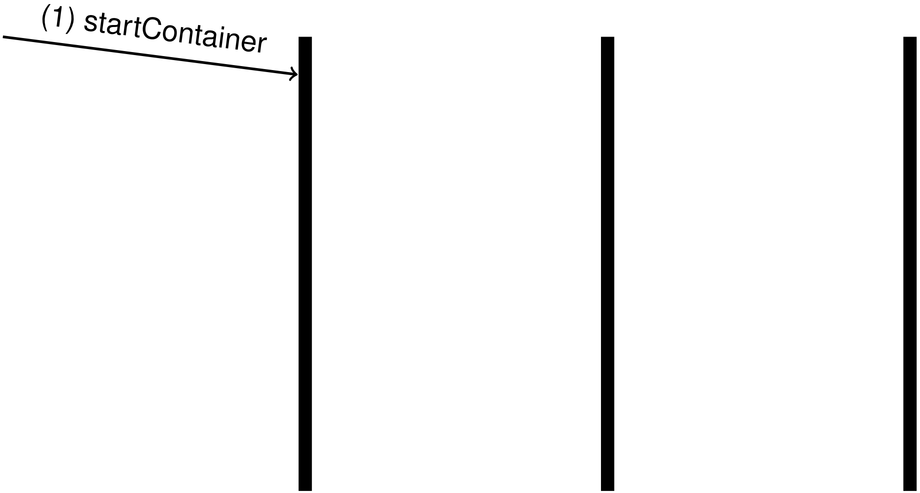


YARN-7786

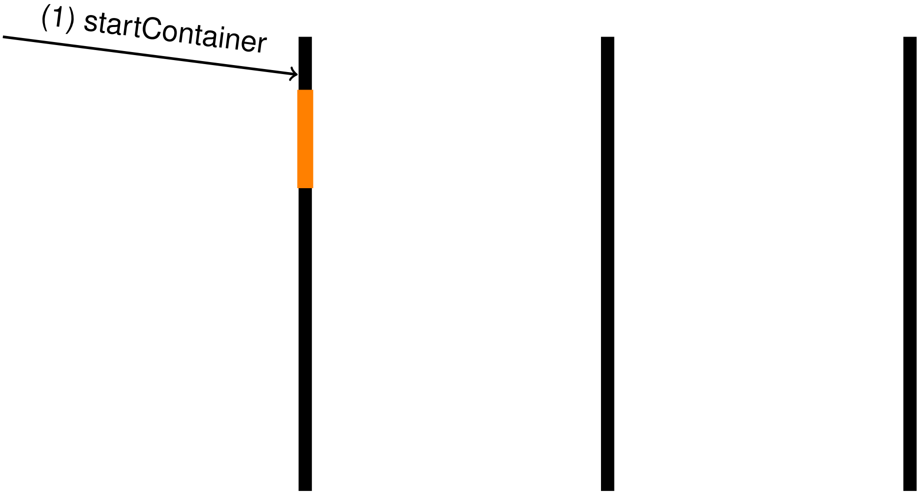


YARN-7176

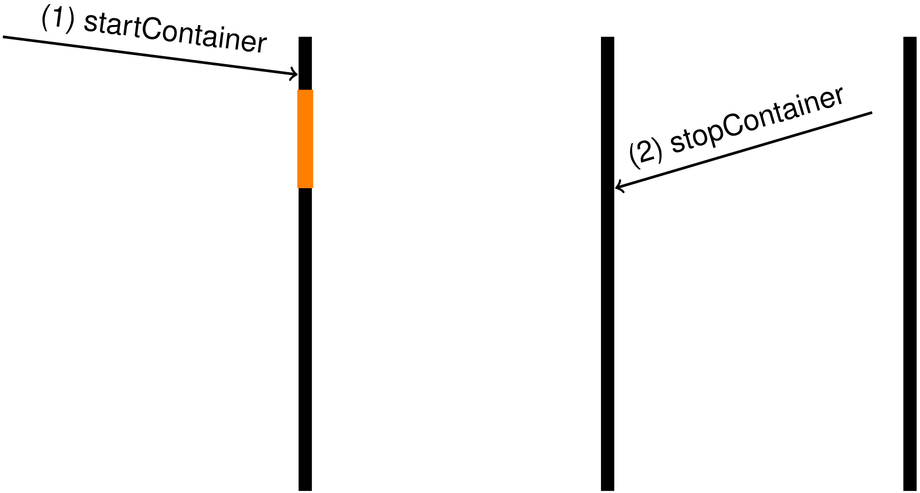
YARN-7176



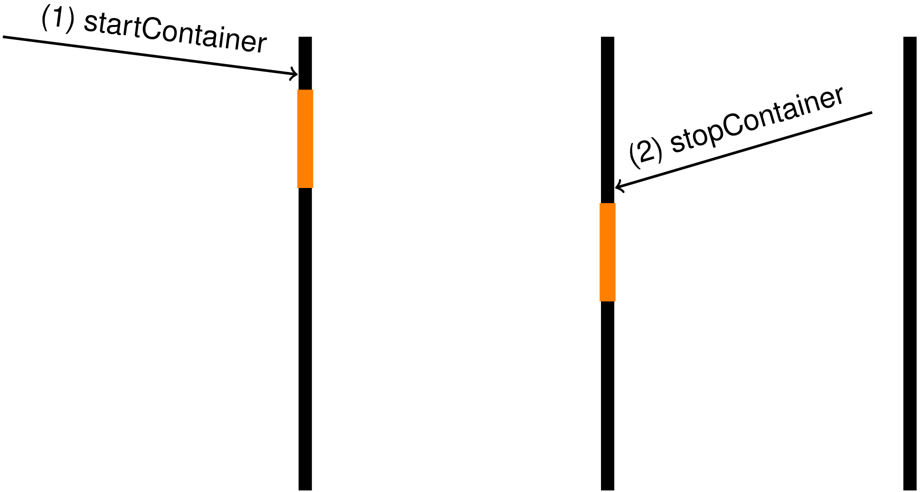
YARN-7176



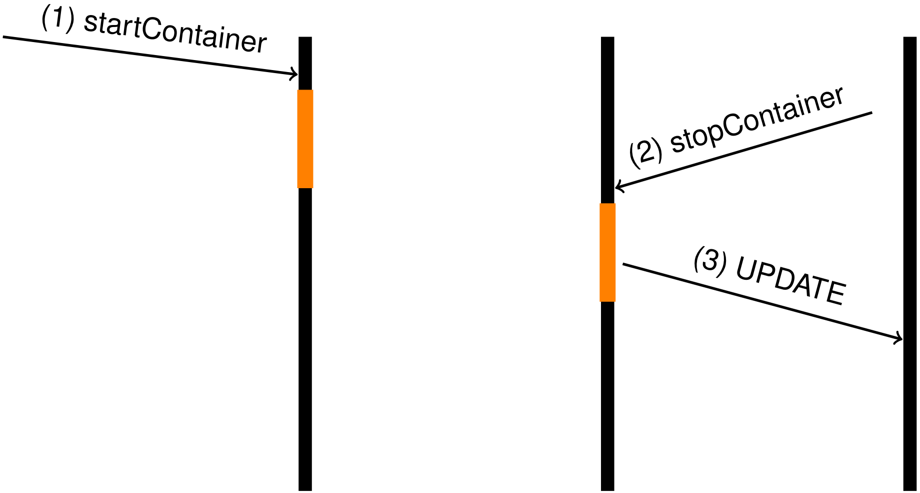
YARN-7176



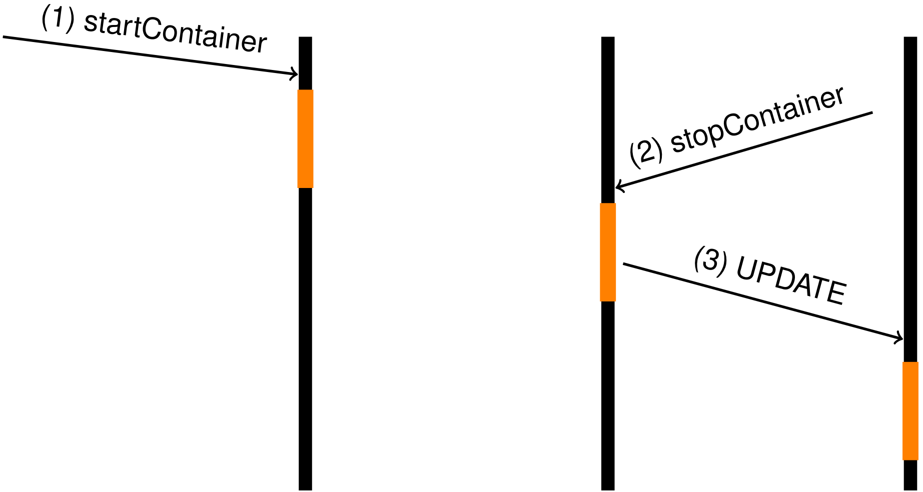
YARN-7176



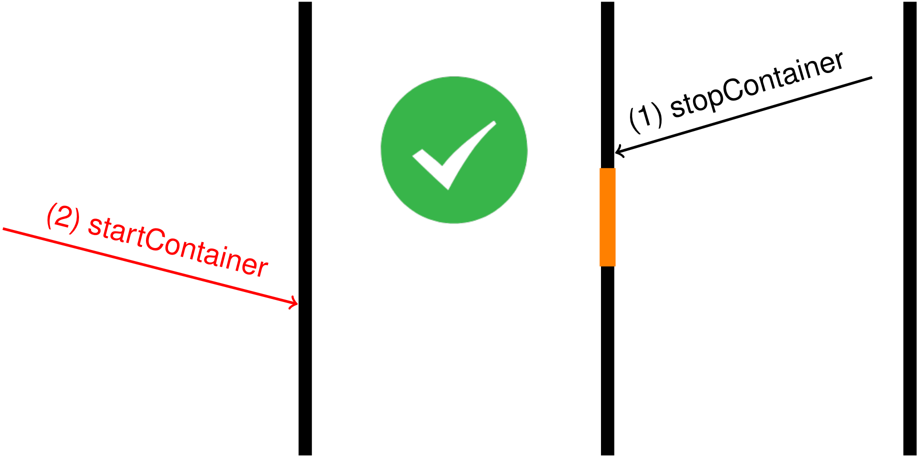
YARN-7176



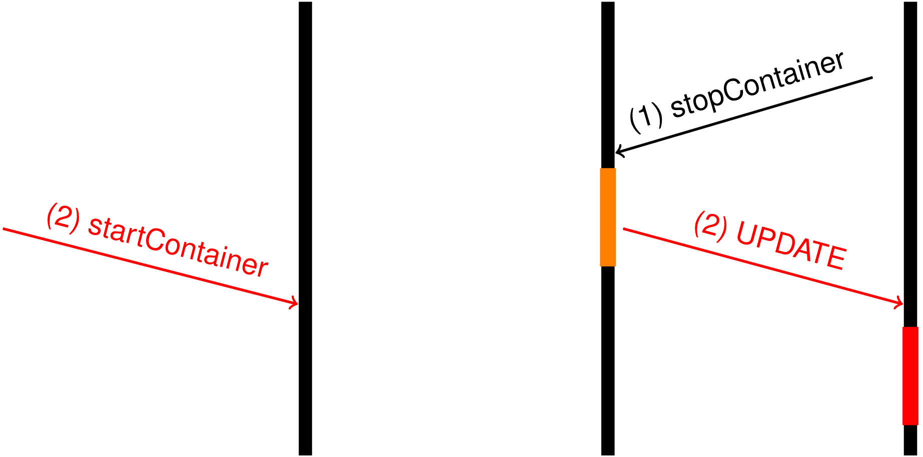
YARN-7176



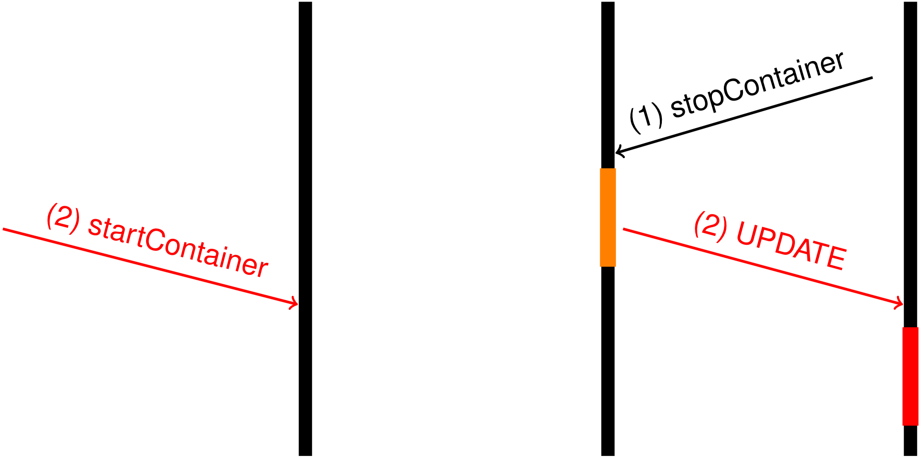
YARN-7176



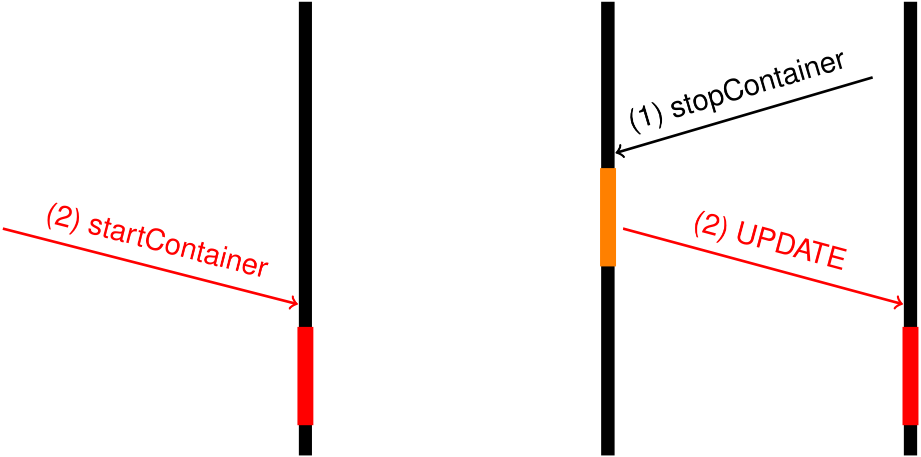
YARN-7176



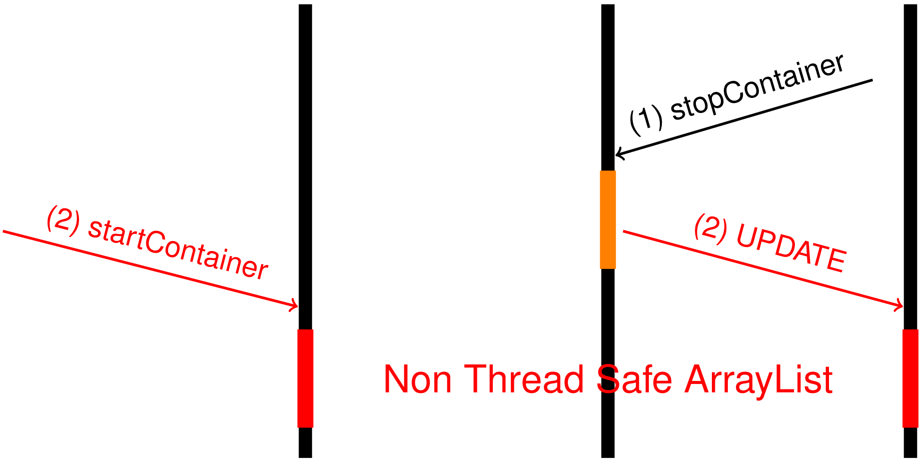
YARN-7176



YARN-7176



YARN-7176



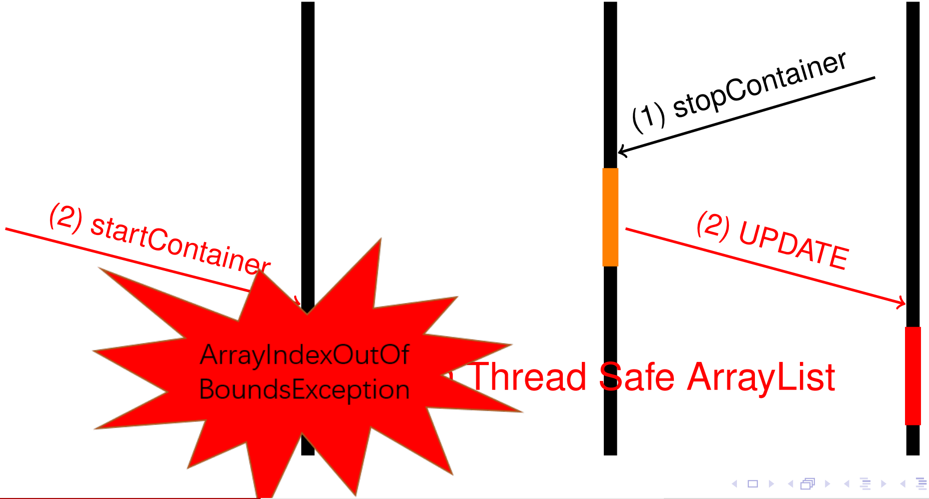
(1) stopContainer

(2) startContainer

(2) UPDATE

Non Thread Safe ArrayList

YARN-7176


(1) stopContainer
(2) startContainer
(2) UPDATE
ArrayIndexOutOf BoundsException
Thread Safe ArrayList

Related work

- Concurrency bug detection
  - Model checking[4]:State-space explosion
  - DCatch[5]:Data Race, memory only;
- Log analysis[6]
  - Diagnose or monitor;
- Study on distributed bugs[7]

[4] **leesatapornwongsa2014samc**.

[5] **liu2017dcatch**.

[6] **xu2009detecting**.

[7] **gunawi3000bugs**; **leesatapornwongsa2016taxdc**.

Conclusions

- CloudRaid:a simple yet effective tool to detect distributed concurrency bugs!

Conclusions

- CloudRaid:a simple yet effective tool to detect distributed concurrency bugs!
- Testing 40 versions of 4 diffrent systems in 35hours!

Conclusions

- CloudRaid:a simple yet effective tool to detect distributed concurrency bugs!
- Testing 40 versions of 4 diffrent systems in 35hours!
- Found 28 bugs, including 8 new bugs!

Thank You

Q & A