FAILURE DETECTION

CAPTURING AND ENHANCING IN SITU SYSTEM OBSERVABILITY FOR FAILURE DETECTION

Peng Huang, Johns Hopkins University; Chuanxiong Guo, ByteDance Inc.; Jacob R. Lorch and Lidong Zhou, Microsoft Research; Yingnong Dang, Microsoft

What constitutes a failure?

What constitutes a failure?

- Crash

What constitutes a failure?

- Crash

- Gray Failure

What constitutes a failure?

- Crash

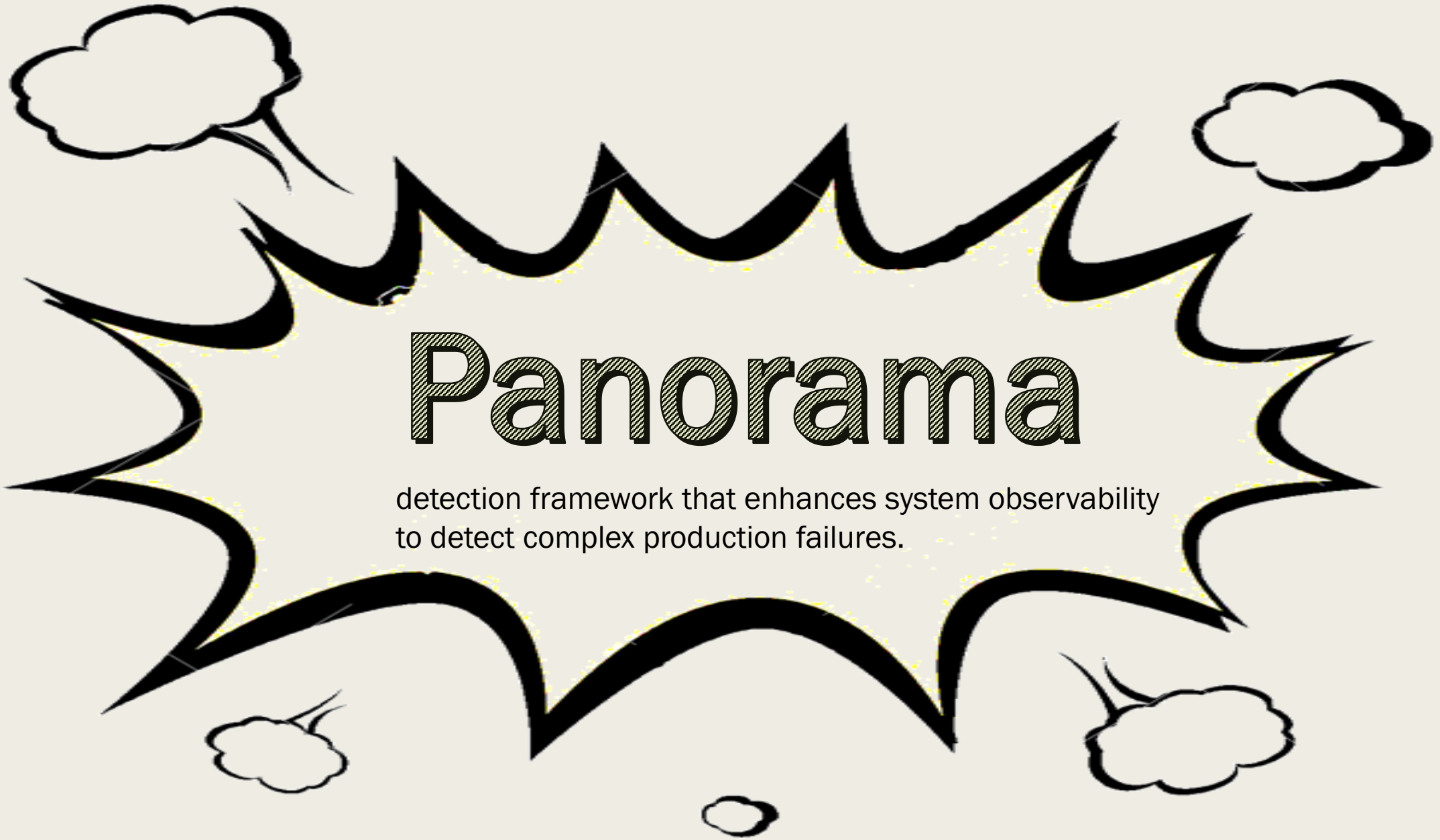
- Gray Failure

Critical thread of a proces got stuck while its other threads including a failure detector keep runnung.


THIS IS FINE.

Why is it so hard to identify failures?

- Components tend to handle errors rather than report them. Due to „separation of concern” principle.
  - *components should hide as much information as they can*
  - *Failure detection should be each component’s own job*
- Even if a component has an incentive to report, it may not have a convenient systematic mechanism to do so.
- Even if a failure is logged, it lacks valuable information about the context because it is analyzed offline.

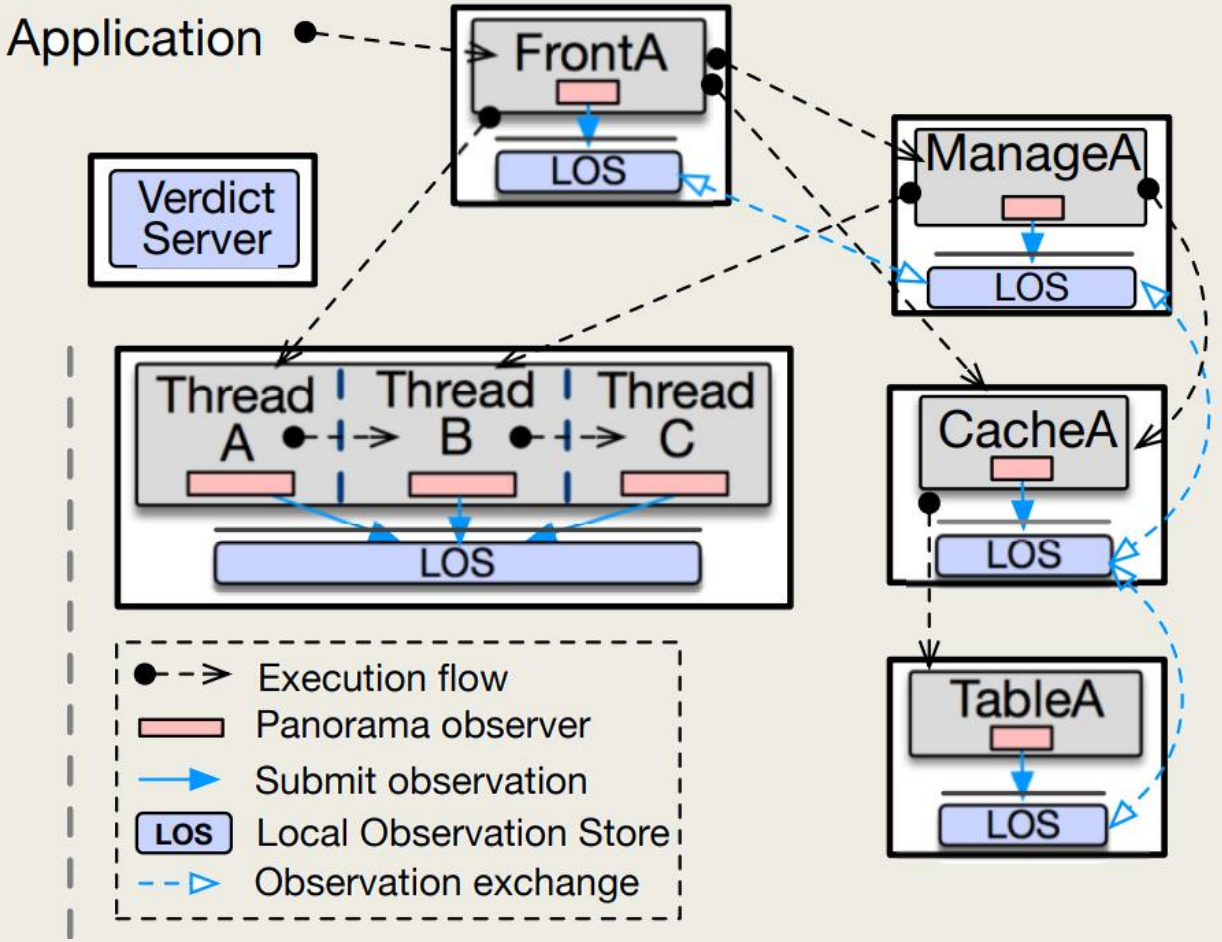

Panorama
detection framework that enhances system observability
to detect complex production failures.

Panorama

- Significantly outperforms existing failure detectors.
- It detects crash failures faster.
- It detects 15 real-world gray failures in less than 7 s each, whereas other detectors only detect one in 86 s.
- Not only detects, but also locates failures.
- Is resilient to transient failures and is stable in normal operations.
- Introduces only minor overhead to the tested systems.
- leverages existing code that lies near the boundaries between different components

How does it work?

- Panorama automatically **inserts hooks into existing code** to submit observations via RPC calls.
- Each process has a Local Observation Store (LOS) that stores all the observations that are made either by or about this process.
- Panorama runs a dissemination protocol to exchange observations among a clique of **LOS**es that share common components.
- Decision engine analyzes observations in LOS and makes a judgment about the process's status



Observations

Observations are done by an **Observer** on a **Subject**

- For each process we have a range of health values.
  - *HEALTHY, DEAD and a few levels of UNHEALTHY*
- An observation contains
  - *a timestamp of when the observation occurred*
  - *the identities of the observer and subject*
  - *the inferred statusof the subject*
  - ***context** describing what the observer was doing when it made the observation*
- A local decision engine analyzes observations to reach a verdict for each subject. This decision result is stored in the verdict table, keyed by subject.

Judging system

Superficial approach:

- Use the latest observation as verdict.

Judging system

Superficial approach:

- Use the latest observation as verdict.

☹ subject experiencing intermittent errors may be treated as healthy

Judging system

Superficial approach:

- Use the latest observation as verdict.

☹ subject experiencing intermittent errors may be treated as healthy

An alternative:

- Reach a verdict if any recent negative observation

Judging system

Superficial approach:

- Use the latest observation as verdict.

☹ subject experiencing intermittent errors may be treated as healthy

An alternative:

- Reach a verdict if any recent negative observation

☹ One biased observer, whose negative observation is due to its own issue, can mislead others.

Judging system

Superficial approach:

- Use the latest observation as verdict.

☹ subject experiencing intermittent errors may be treated as healthy

An alternative:

- Reach a verdict if any recent negative observation

☹ One biased observer, whose negative observation is due to its own issue, can mislead others.

Bounded look-back majority

- Group the observations by the unique observer
- To reach a final verdict for each context across all groups, the summaries from different observers are aggregated and decided based on a simple majority

Locate Observation Boundary

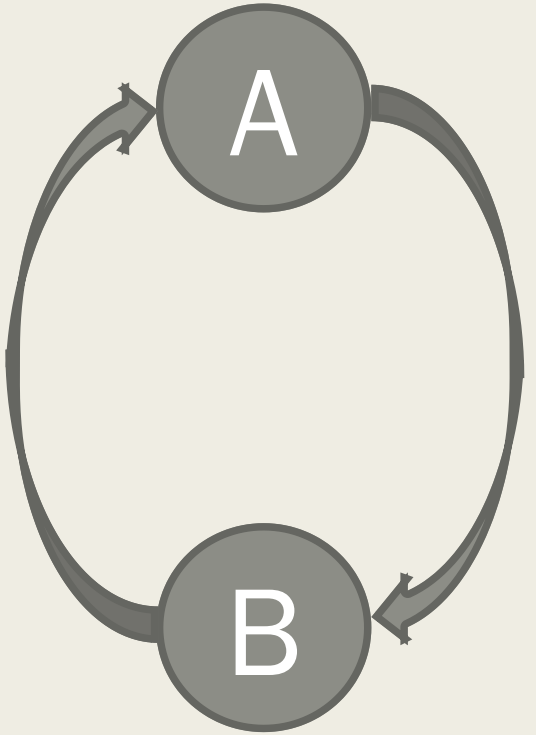
Panorama only extracts errors generated when crossing component boundaries, because these constitute observations from the requester side.

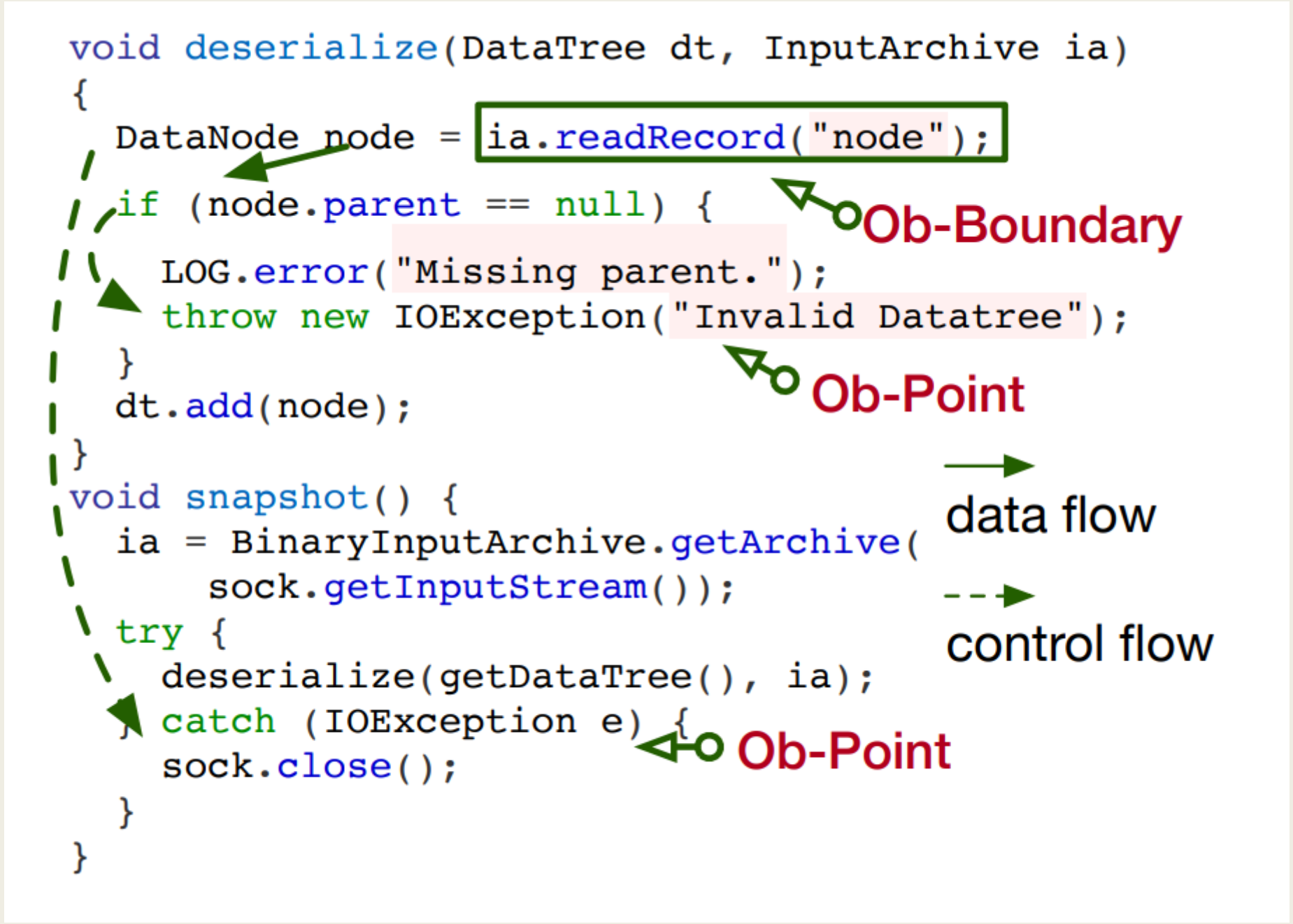
- inter-process boundary
  - *library API invocation,*
  - *a socket I/O call,*
  - *a remote procedure call (RPC).*

- inter-thread boundary
  - *call crossing two threads within a process*

Inserting hooks

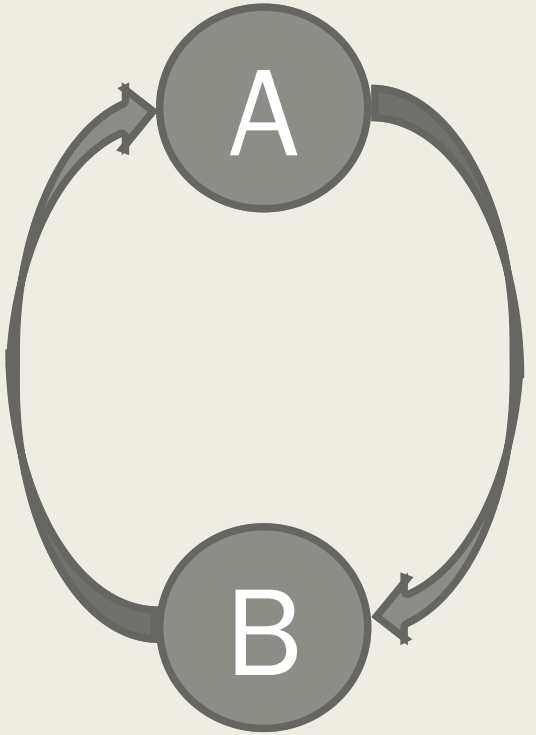
```
try {
   firstProcessor.processRequest(si);
} catch (RequestProcessorException e)
{
   LOG.error("Unable to process
request: " + e);
}
```




void deserialize(DataTree dt, InputArchive ia)
{
DataNode node = ia.readRecord("node");
if (node.parent == null) {
LOG.error("Missing parent.");
throw new IOException("Invalid Datatree");
}
dt.add(node);
}
void snapshot() {
ia = BinaryInputArchive.getArchive(
sock.getInputStream());
try {
deserialize(getDataTree(), ia);
} catch (IOException e) {
sock.close();
}
}
Ob-Boundary
Ob-Point
Ob-Point
data flow
control flow

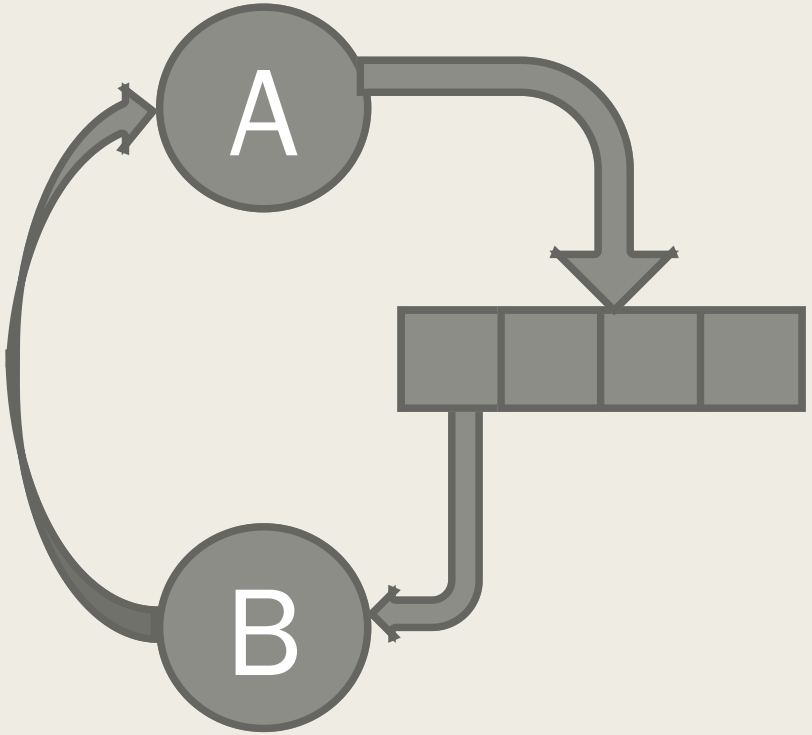
Inserting hooks

```
try {
   firstProcessor.processRequest(si);
} catch (RequestProcessorException e)
{
   LOG.error("Unable to process
request: " + e);
}
```

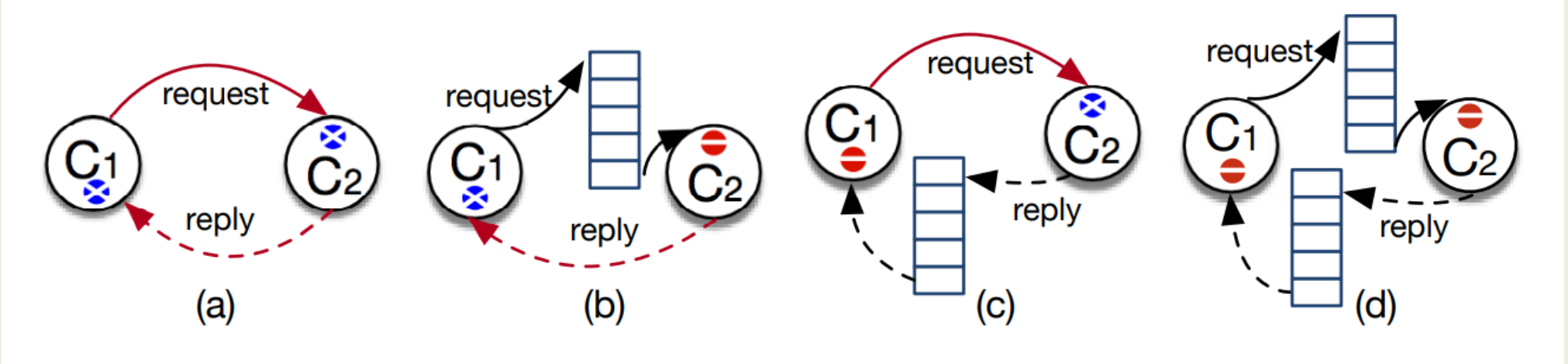


Inserting hooks

```
try {
   firstProcessor.processRequest(si);
} catch (RequestProcessorException e)
{
   LOG.error("Unable to process
request: " + e);
}
```



4 Cases



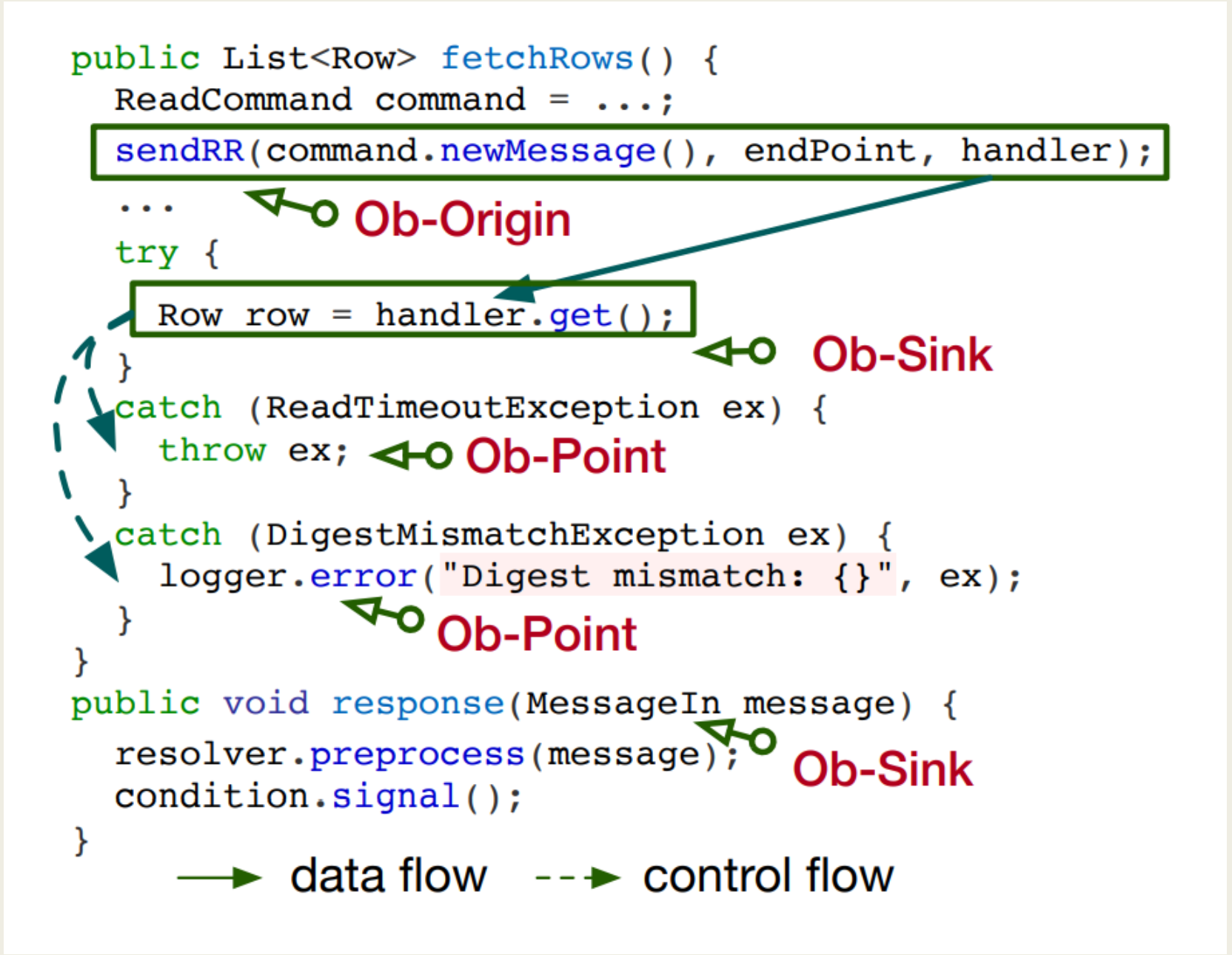
failure is **unobservable** to the other component

failure is **observable** to the other component

Handling indirection

indirection splits a single interaction between components among multiple observation boundaries.

- When analyzer finds an **ob-origin**, it inserts a hook that submits an observation with the special status PENDING.
- At any **ob-sink** indicating positive evidence, analyzer inserts a hook to report a HEALTHY observation.
- At any **ob-sink** indicating negative evidence, the analyzer inserts a hook to report a negative observation


public List<Row> fetchRows() {
ReadCommand command = ...;
sendRR(command.newMessage(), endPoint, handler);
...
try {
Row row = handler.get();
}
catch (ReadTimeoutException ex) {
throw ex;
}
catch (DigestMismatchException ex) {
logger.error("Digest mismatch: {}", ex);
}
}
public void response(MessageIn message) {
resolver.preprocess(message);
condition.signal();
}
Ob-Origin
Ob-Sink
Ob-Point
Ob-Point
Ob-Sink
data flow
control flow

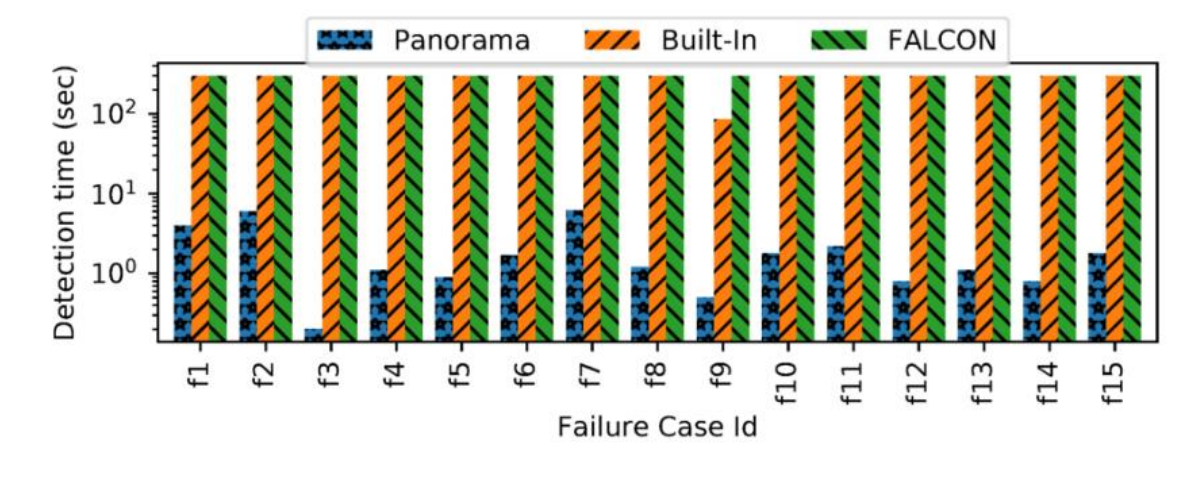
Is it any good?

Detection of Crash Failures

Detector	Crash Failure Injection Site							
	ZooKeeper		Cassandra		HDFS		HBase	
	leader	follower	seed	datanode	namenode	datanode	master	regionserver
Built-in	13 ms	3 ms	28 s	26 s	708 ms	30 s (12 min*)	11 s	102 ms
Panorama	8 ms	2 ms	8 s	9 s	723 ms	6 s	1.5 s	102 ms

- We see that with Panorama the observers take less than 10 s to detect all ten cases, and indeed take less than 10 ms to detect all ZooKeeper failures.

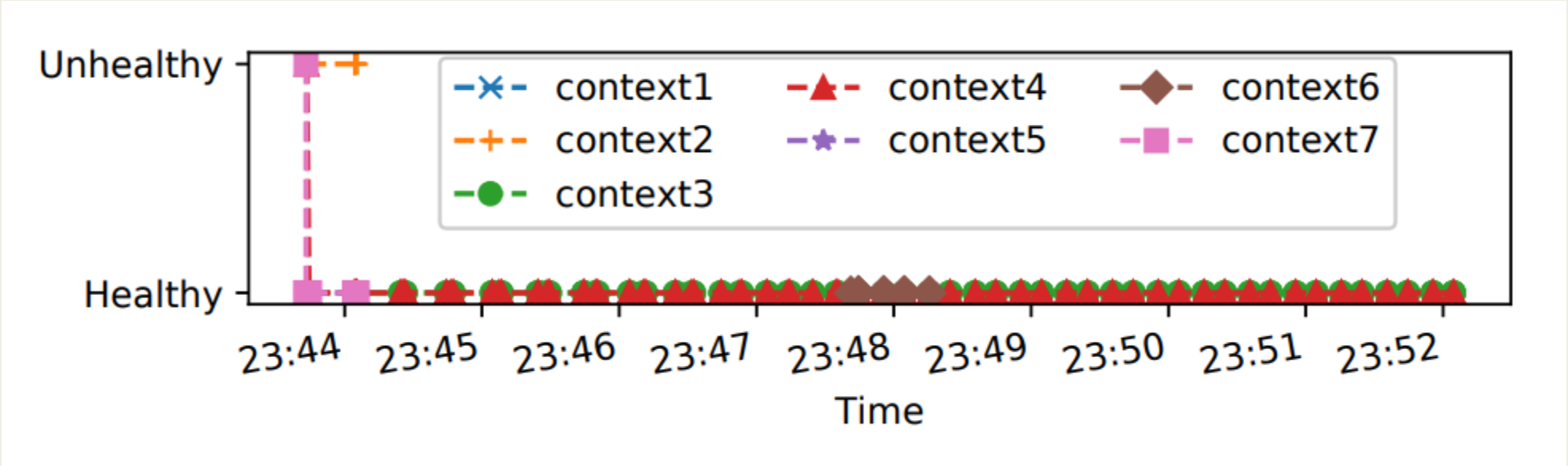
Detecting gray failures



ID	System	Fault Synopsis
f1	ZooKeeper	faulty disk in leader causes cluster lock-up
f2	ZooKeeper	transient network partition leads to prolonged failures in serving requests
f3	ZooKeeper	corrupted packet in de-serialization
f4	ZooKeeper	transaction thread exception
f5	ZooKeeper	leader fails to write transaction log
f6	Cassandra	response drop blocks repair operations
f7	Cassandra	stale data in leads to wrong node states
f8	Cassandra	streaming silently fail on unexpected error
f9	Cassandra	commitlog executor exit causes GC storm
f10	HDFS	thread pool exhaustion in master
f11	HDFS	failed pipeline creation prevents recovery
f12	HDFS	short circuit reads blocked due to death of domain socket watcher
f13	HDFS	blockpool fails to initialize but continues
f14	HBase	dead root drive on region server
f15	HBase	replication stalls with empty WAL files

Gray Failure #1

Transient Failures



Performance

System	Latency		Throughput	
	Read	Write	Read	Write
ZK	69.5 µs	1435 µs	14402 op/s	697 op/s
ZK+	70.6 µs	1475 µs	14181 op/s	678 op/s
C*	677 µs	680 µs	812 op/s	810 op/s
C*+	695 µs	689 µs	802 op/s	804 op/s
HDFS	51.0 s	61.0 s	423 MB/s	88 MB/s
HDFS+	52.5 s	62.2 s	415 MB/s	86 MB/s
HBase	746 µs	1682 µs	1172 op/s	549 op/s
HBase+	748 µs	1699 µs	1167 op/s	542 op/s

References

- All graphs tables and listings were taken from the aforementioned paper.