The workflow motif: a powerful abstraction for debugging distributed systems

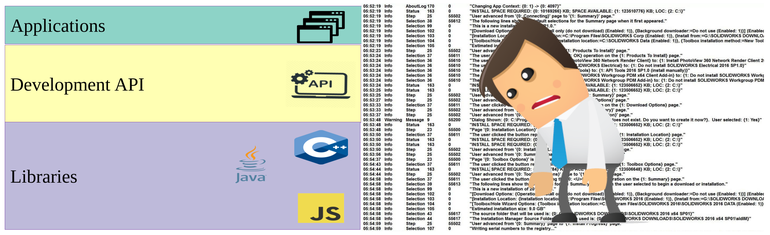
Mania Abdi, Golsana Ghaemi, Mark Crovella, Orran Krieger, Peter Desnoyers, Ata Turk, Raja Sambasivan

Northeastern

BU

Abstractions for Building vs. Debugging Production Systems

- Abstractions enable developers to build complex distributed systems
  - **Build complex systems from simpler building blocks**
  - **Hide details of implementation.**

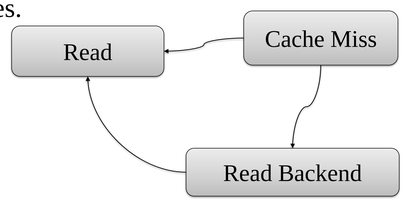


- De facto approach to debugging is to use no abstractions, whatsoever.
  - **i.e. Logs of low level events**

Extreme mismatch between building systems vs debugging them: key reason why diagnosis is so challenging

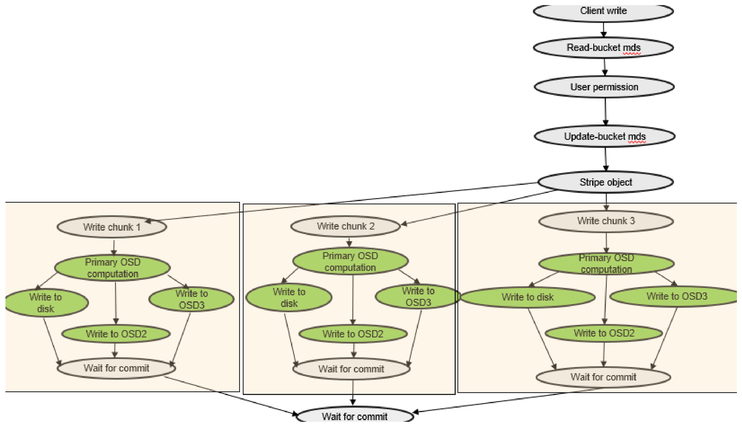
Workflow Motifs: a Novel Diagnosis Abstraction

- Graphs that describe frequent processing patterns in the workflow of how requests are processed along with their performance.
- Building blocks of distributed systems runtime behavior.
  - Examples: Behavior , leader election in consensus protocol.
- Mitigate complexity during diagnosis by:
  - Allowing problems to be understood in terms of behavioral building blocks.
  - Allow the details of irrelevant problem to be hidden.
- Enables new diagnosis use cases.



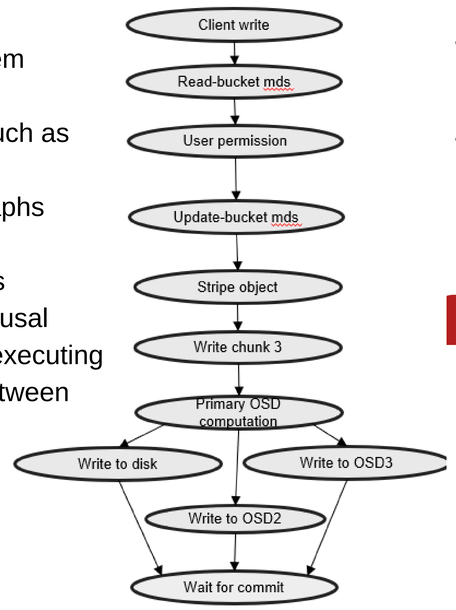
Example of write workflow in Ceph and its motif

- Example Motif: Three way replicatio



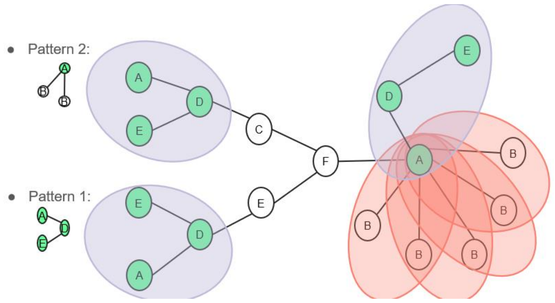
Key enabler one: E-2-E tracing

- Is a way of capturing how each request is processed among different components of distributed system
- Stores information of executing
- Carries trace/request context such as trace ID
- Traces are Directed Acyclic Graphs (DAG)
  - Nodes are trace points
  - Edges represent the causal relationships between executing process and latency between them



Key enabler two: Frequent subgraph mining algorithms

- Used in Biology for DNA matching, chemistry for component matching.
  - Example: gaston, pafi
- Used to mine e-2-e traces for frequent patterns
- Way they work:
  - Find similar subgraphs of different size within a database of graphs.
  - Count the # of occurrence of each subgraph



UseCases

- Contrast and compare request workflows
  - Fining common motifs with in each execution and compare their structure.
- Improve Slow performance
  - Identify slowest motifs and present them to engineers so that they can optimize them.
- Identify anomaly
  - Identify request containing motifs that usually don't occur together

Current Status & Open Questions

- How should we modify the frequent subgraph mining algorithms to be less expensive?
- What properties tracing infrastructure should support to capture workflow motifs?
- How can we take advantage of domain knowledge?
- What other use cases motifs can be useful for?
- Enable state-of-art tracing infrastructure on a complex storage system (CEPH)
- Explore different frequent subgraph algorithm on UrsaMinor Traces and HDFS traces.