Event-Sourced Monitoring of Your HTCondor Cluster

Kevin Retzke
HTCondor Week
23 May 2019
“Traditional” Sample-Based Monitoring

• Collect metrics (e.g. how many jobs are running) at regular intervals
  – Historical trends
  – Throughput
  – Usage by user
  – Health
• You already do this

• … Right?
What happens between samples?

A Lot!
Event-Based Monitoring

• Event Sourcing: collecting and storing every change to the state of a system instead of or in addition to storing the current state.
  – “realtime” data with minimal collection lag. Collecting thousands of metrics for hundreds of thousands of jobs can take a while.
  – “infinite” granularity, down to the precision of your timestamps (I can has millis?).
  – Numerous open-source tools for working with event data, e.g.
    • Kafka https://kafka.apache.org/
    • Spark Streaming https://spark.apache.org/streaming/
    • Faust https://faust.readthedocs.io/en/latest/
  – State can be determined at any point of time…
Tracking State

… if you have the state corresponding to some exact known point in your events.

… and you aren’t missing any events.

...let’s focus on using events directly

(for now – there are some interesting tools in this area, e.g. https://eventstore.org/ that I want to explore more)
Use Case: “Blackhole” Node Detection

- Fact: computers break
- How can we detect a bad worker node (often at another site*), that is causing jobs to fail, and stop sending jobs there before it sucks up the entire queue (hence “blackhole”)?
- Events provide the perfect data set to monitor for blackholes.
  - Lots of failing jobs
  - No successful jobs
  - Held jobs
  - Shadow exceptions
  - Disconnections
  - No events

* But never at UW
Monitor in Grafana

Send alerts to Slack (or email, or ticket, etc)
Use Case: Is My Submission Done Yet?

- How do you quickly determine the status of hundreds of submissions (a cluster or DAG) with thousands of jobs each, as fast as a user can push F5, without overwhelming your schedds?

- Count the events:
  \[
  \text{SubmitEvents} \leq \text{JobTerminatedEvents} + \text{JobAbortedEvents}
  \]

- Or if you want to consider it done when all the jobs are terminated or held:
  \[
  \text{SubmitEvents} \leq \text{JobTerminatedEvents} + (\text{JobHeldEvents} - \text{JobReleaseEvents}) + \text{JobAbortedEvents}
  \]
HOWTO: Enable in HTCondor

• Enable global event log in schedd, just set the path and file name:

\[
\text{EVENT\_LOG} = /var/log/condor/EventLog
\]

• Add additional ClassAd attributes (optional, but recommended, and required for our logstash config):

\[
\text{EVENT\_LOG\_JOB\_AD\_INFORMATION\_ATTRS} = \text{Owner} \ \text{DAGManJobId} \ \text{MachineAttrMachine0} \ \text{JobCurrentStartDate}
\]

  – Note that this adds a second “information” event for every trigger event.

• May need to add machine attributes to job ClassAds:

\[
\text{SYSTEM\_JOB\_MACHINE\_ATTRS} = \text{Machine}
\]

• Job event log code reference:

Sample Event

Job ID  Timestamp


... 028 (18938569.000.000) 05/20 12:14:51 Job ad information event triggered.
Proc = 0
MachineAttrMachine0 = "fnpc7212.fnal.gov"
EventTime = "2019-05-20T12:14:51"
TriggerEventTypeName = "ULOG_EXECUTE"
Jobsub_Group = "sbnd"
MachineAttrGLIDEIN_Site0 = "FermiGrid"
TriggerEventTypeNumber = 1
JobCurrentStartDate = 1558372490
MyType = "ExecuteEvent"
Owner = "aezeribe"
MachineAttrGLIDEIN_ResourceName0 = "GPGrid"
Cluster = 18938569
Subproc = 0
EventTypeNumber = 28
...
HOWTO: Collect Events

• Logstash: Swiss Army Knife of data
  – https://www.elastic.co/products/logstash
  – Config: https://github.com/fifemon/logstash-config/blob/master/condor.logstash.conf

• File input
  
  path => "/var/log/condor/EventLog"

• Split events
  
  delimiter => "

  ...

  "

• Combine multiple lines: any line that doesn’t begin with a number belongs to the previous event.

  codec => multiline {
    pattern => "^[^\d]"
    what => "previous"
  }
HOWTO: Process events

- Grok filter to match events

```ruby
match => {
  "message" => [
    "%{CONDOR_EVENT:event}
    %{DATA:event_message}\n%{GREEDYDATA:event_body}'",
    "%{CONDOR_EVENT:event} %{DATA:event_message}"
  ]
}
```

- Grok patterns to get job ID and timestamp from each event

```
CONDOR_TIMESTAMP %{MONTHNUM}/%{MONTHDAY} %{TIME}
CONDOR_EVENT %{INT:event_code}
\(%{INT:cluster:int}\.%{INT:process:int}\.%{INT:subprocess:int}\)
%{CONDOR_TIMESTAMP:condor_timestamp}
```

- [https://github.com/fifemon/logstash-config/blob/master/patterns/condor](https://github.com/fifemon/logstash-config/blob/master/patterns/condor)
HOWTO: Combine Events

- **Aggregate filter: Save trigger event**
  
  ```
  task_id => "%{cluster}.%{process}.%{subprocess}"
  code => "map['trigger_event_message']=event['message']"
  map_action => "create"
  ```

- **Aggregate filter: Add trigger event to information event**
  
  ```
  task_id => "%{cluster}.%{process}.%{subprocess}"
  code => "event['trigger_event_message']=map['trigger_event_message']"
  map_action => "update"
  end_of_task => true
  timeout => "60"
  ```

- **Grok patterns to pull interesting fields from trigger event**
  
  ```
  match => {
    "trigger_event_message" => [ 
      "%{CONDOR_EVENT_001}",
      "%{CONDOR_EVENT_006}",
      ...
  }
  ```
HOWTO: Store and Analyze Events

• Store in Elasticsearch

```
Output {
  elasticsearch {
    hosts => [ "localhost:9200" ]
    index => "condor-events-\{+YYYY.MM\}"
  }
}
```

• Analyze in Kibana and Grafana
Holistic HTCondor Monitoring

- Events
- Data Transfers
- Snapshot Metrics
- Raw ClassAds
Other Parts of Holistic Monitoring at Fermilab

- Snapshot metrics to time-series database
  - [https://github.com/fifemon/probes](https://github.com/fifemon/probes)
  - (several forks with different features, some efforts to merge)
- Job history collection to elasticsearch with filebeat and logstash
- Raw classad collection to elasticsearch with condorbeat
  - [https://github.com/retzkek/condorbeat](https://github.com/retzkek/condorbeat)
- Data transfers – very little through HTCondor itself
  - Client log (IFDH) through rsyslog to elasticsearch with logstash
  - dCache transfer history to elasticsearch with logstash
- Everything routed through Kafka for resilience, replaying, testing, etc.
Kevin Retzke | Event-Sourced Monitoring of Your HTCondor Cluster