2018-06-06 OW Tech Interchange - Meeting Notes

Notes:

- Tyson Norris is moderating today
- last meeting was 2018-05-23
- Attendees:
    - Michael Marth, Sam Baxter, Tyson, Priti, Martin, Sandeep, James Thomas, Rodric, Dan, Dragos, Vijay, KeonHee Kim, Neeraj, Mparuthickal (Matthew), Chetan, Ramki, Himavanth, Dominic, Olivier, Chris, Justin, Vadim, Carlos, Tzuchiao, Duy

Introductions of new attendees

- Mparuthickal (Matthew) - NYC, last 4-5 months been exploring OW, delivering all data fabric via OW api
- Viay - also trying to inc. OW in our environment, doing PoC using OW last 2 months
- Sam - 2nd call, Phd student at UMass an IBM intern, looking at long-running computations using javascript

Open comments on status/updates in a few areas:

- Main/core OpenWhisk (Carlos/Jeremias/Markus/Tyson)
    - https://github.com/apache/incubator-openwhisk/pulls?utf8=%E2%9C%93&q=is%3Apr+is%3Amerged
    - Status updates:
        - Runtime manifest stem cell config.
            - now can provide pre-warm configs for each kind in runtime manifest
        - one big problems with pre-warm was not being able specifying diff. configs for different “kinds”
    - Other PRs:
        - Tyson: nothing much notable from my look into PRs
        - Carlos will provide later in agenda if time
- **Scheduling algorithm change (Dominic)**
    - (Shares slides):
    - Shows Kafka consumer log and message processing pipeline
        - Discusses autonomous container scheduling “basics”
            - shows how container scheduling from kafka->Invoker focusing on how # consumers and # partitions are currently handled.
            - # of partitions = # of concurrent containers (today)
        - Shows diagram of Invoker status via “health” message
        - Shows “Even" load distribution slide
        - Shows “Performance Isolation” slide
        - Shows “Improved Base TPS” slide
            - tests show 1,500 TPS for 1 container
        - Deterministic TPS slide
            - Total TPS for Action A: 4,500
            - for 6,000 TPS -> Add one more container
        - Goal is deterministic / predictable TPS
            - In this proposal Invokers read from kafka topiccs
            - Tyson: how do container get deleted in your proposal?
            - Dom: if no request for some time (e.g., 10 sec, configurable) then deletion happens
            - Tyson: by who
            - Dom: container proxy?
            - Tyson: it commits “suicide”?
            - Dom: effectively yes
            - Tyson: if container proxy resp. for killing itself, you could run out of resources, because they may never kill themselves… controller needs to be involved?
                - can take this offline…
            - Dom: shows “worst case” “does not wait for completion” slide
            - Dom: shows Pros/Cons of proposal
                - Pros:Container reads request from kafka, request processing not affected by container creation/deletion
                - Cons: consumer lag for each request -> increase exec. time
                    - same # of topic with # of Actions (1 topic per action)
                    - Action container can be reserved for 30sec to 1min per action
                    - all runtimes would need to include kafka client
                - Tyson: consumer lags is a concern
                    - checking lag on every request is LOTS of overhead
                - Dom: I did some benchmark to chk. consumer lag… with 10 virtual resources the mean test time was 1ms
                - Tyson: ok, the other issue I had was # of topics is unbounded, (per action) partitions/backing for kafka could become an issue
                - Dom: kafka topics has config. issues… hope is shared topics is possible, but if each topic is unique, the hope is kafka would be able to delete topics if not fully utilized based upon retention configuration
                - Tyson: data not the problem, but consumers having to check for data and brokers need to manage topics is the larger problems… if we approach a million actions…
                - Dom: ran tests upto 1000 topics (30,885 Kafka TPS)
                    - 1k active topics = 1k concurrent containers = 62.5 invokers (8 cores, 10GB memory, max pool size-16)
                    - Matt: appears growth is deterministic (scale out) based upon # of topics
                    - Dom: would need more than 3 nodes of kafka
                - Tyson: data size is manageable, don’t know if broker and consumer will not have resource overhead just having a topic exist at all.
                - Tyson: would like kafka person to comment...

        - Carlos: a basic question, is the consumer/client per unique action? or 1 consumer for 1 invoker?
        - Dragos: some questions
            - if multiple partitions for mult. consumers, we have problems with people reading same message from diff partitions
            - one container does not know how other containers work on same post. containers running at different rates could cause problems
        - Dom: mult partitions with same journal might cause issues
        - Tyson: if consumers for an action topic have same consumer group, then 1 part consumer will receive messages from a part. partition and other consumers will not receive messages from other partitions.
            - what is impact of changing # of partitions “in flight” might cause sig. issues..
        - Don: Shows slide of “Kafka consumer group”
            - shows how partitions (TopicA) are mapped to different Consumer Groups (before/after proposal)
            - Maximum parallelism is limited to the # of partitions
        - Tyson: need to continue this on the “dev” list
        - Carlos: does this assume same kafka cluster? Could be diff kafka clusters for consumers, for mgmt.?
        - Dom: can separate if we want…
        - Tyson: can separate, larger concern is potential or overwhelming the data plane…
        - Dom: shows Perf. evaluation slides
            - BMT Environment show (3 controllers, 3 kafka, 10 invokers, 100 containers
            - Shows TPS increase by # of actions vs. # of containers (avail)
        - Tyson: ***presentation is on CWiki, please review and take it to “dev” list***
    - **Tracing support (Sandeep)**
        - (shares slides):
            - PR 2282 represents this feature, markus and others have been reviewing
                - “Implementation details” - based upon **OpenTracing API**
            - How to enable: Start Zipkin server (docker run), very easy
        - Shows demo...
            - Goes to Zipkin UI and shows Action trace by Controller
            - shows timings of each invocation (with component breakdown)
        - Some issues with cache in zipkin, so may need to refresh (and note on UI that some actions are “still in progress”)
        - Dom: consider using kafka headers for trace context?
        - Sandeep: much easier to use activation messages as trace context, kafka would add to impl. complexity
            - other libs. that help support kafka tracing are not supported on many platforms, this was much easier.
        - Justin: can you show again how this was setup?
        - Sandeep: add application.conf to connect to zipkin backend and use docker to start zipkin server
        - Rodric: how big is context?
        - Sandeep: small object with 3 IDs, very small
        - Rodric: can this be for end user or operators?
        - Sandeep: mainly for operators, look for bottlenecks
        - Rodric: have you thought of incorporating existing system metrics? container reuse etc.?
        - Sandeep: focus is request insight not component insight
        - Tyson: with metrics hard to see details of a particular activation
        - Chetan: in future, in production, we likely do not want to enable this for everything all the time, but add as needed to look at particular customer
        - Rodric: there is already a header (in Nginx) that might help with that

- **Release process**: (Matt/Vincent)
    - Matt updates on source release
        - document and support making JDK configurable
        - All code that can/needs an ASF license now has it
        - Scancode(.py) and Apache RAT (tools) configs. now enforce our policies that are published on the "Release" website
- **Mesos/Splunk update**: (Dragos/Tyson)
    - No update
- **Kubernetes**: (Dave Grove)
    - https://github.com/apache/incubator-openwhisk-deploy-kube/pulls?utf8=%E2%9C%93&q=is%3Apr+is%3Amerged
    - No update
- **API Gateway** (Matt Hamann/Dragos)
    - https://github.com/apache/incubator-openwhisk-apigateway/pulls
    - No update
- **Catalog/Packages/Samples** (anyone)
    - No update
- **Tooling/Utilities** (Carlos/Matt/Priti)
    - **Wskdeploy**
        - Priti updates on latest wskdeploy changes/enahncements/fixes
            - Parameter passing (command line > Depl. file > Environment (applied to) Project and cascades to low level entities (e.g., Actions)

Confirm moderator for next call

- **Dragos** will volunteer, for June 20th meeting
- adjourn 11:01 AM US Central