ArgoCon

NORTH AMERICA

Argo Workflows submissions latency optimization

ArgoCon
NORTH AMERICA

November 12, 2024
Salt Lake City

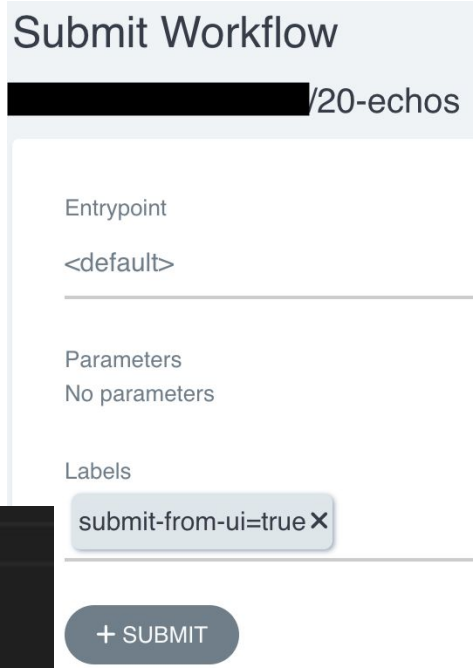
Greg Pomykala
Principal Software Engineer
Splunk (a Cisco Company)

What workflow submission really takes

- Argo cli invocation
- Argo Server API call (api client or Argo UI)
- POSTing a workflow CR with k8s client
- Argo SDK
- Argo Events
- GitOps

```
// Submit the workflow
workflowService := client.NewWorkflowServiceClient()
createdWorkflow, err := workflowService.CreateWorkflow(context.TODO(),
    Namespace: "argo", // Replace with your namespace
    Workflow:  workflow,
})
```

```
1  argo submit 20-echos.yml
2
3  kubectl create -f 20-echos.yml
4
5  curl 'https://my-argo-server-url/api/v1/workflows/my-namespace/submit' \
6    -H 'content-type: application/json' \
7    -H 'cookie: authorization="Bearer XXX"' \
8    --data-raw '{"namespace":"stack-orchestration","resourceKind":"Workflow
```



How long does it take?

100ms, 1s, 1m, 10m?

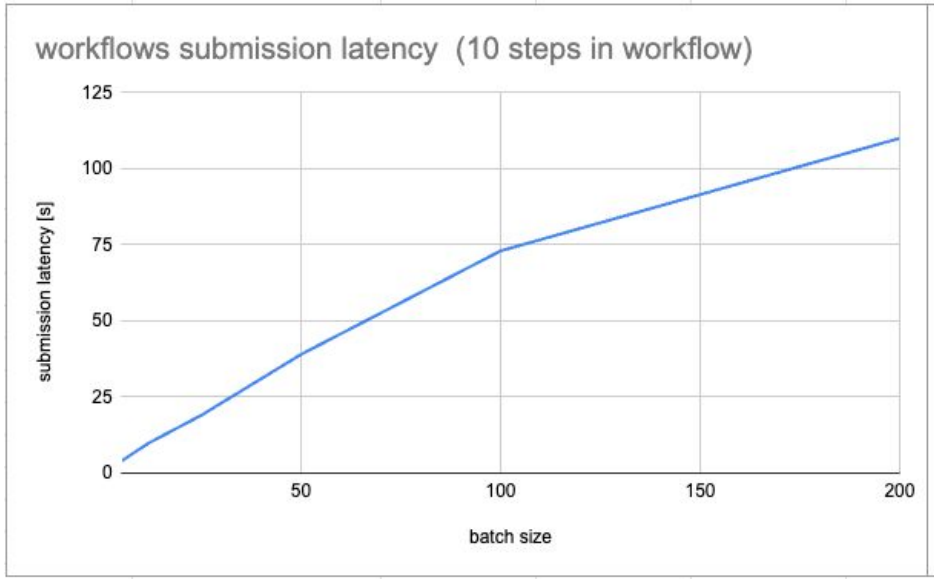
```
apiversion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  name: {{ steps_cnt }}-echos
spec:
  templates:
    - name: start
      dag:
        tasks:
{% for i in range(1, steps_cnt + 1) %}
          - name: echo-{{ i }}
            templateref:
              name: echo
              template: print-message
{% endfor %}
  entrypoint: start
```

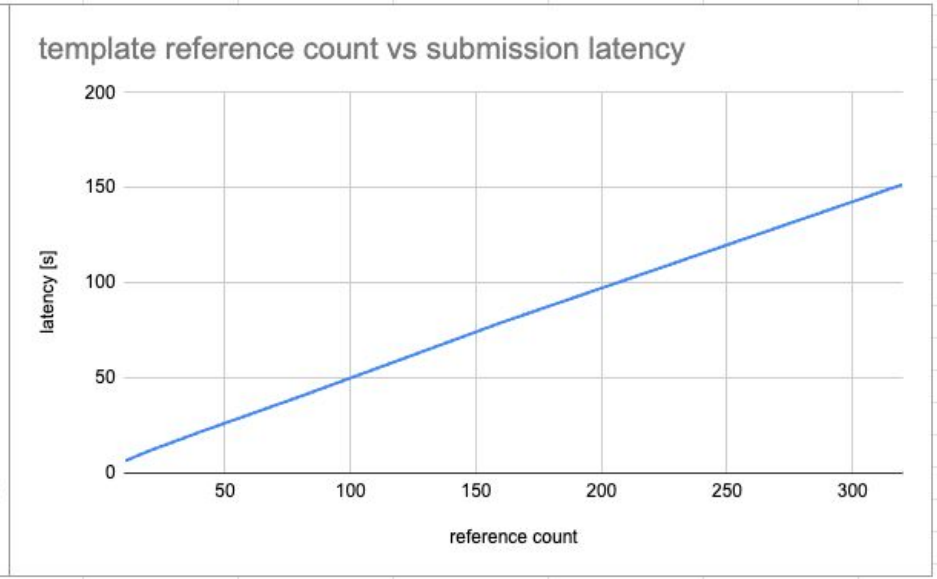
```
time argo submit wf.yml
```

```
jinja2 workflow-template.yaml -D steps_cnt=10 | yq eval > wf.yml
```

What does it depend on?

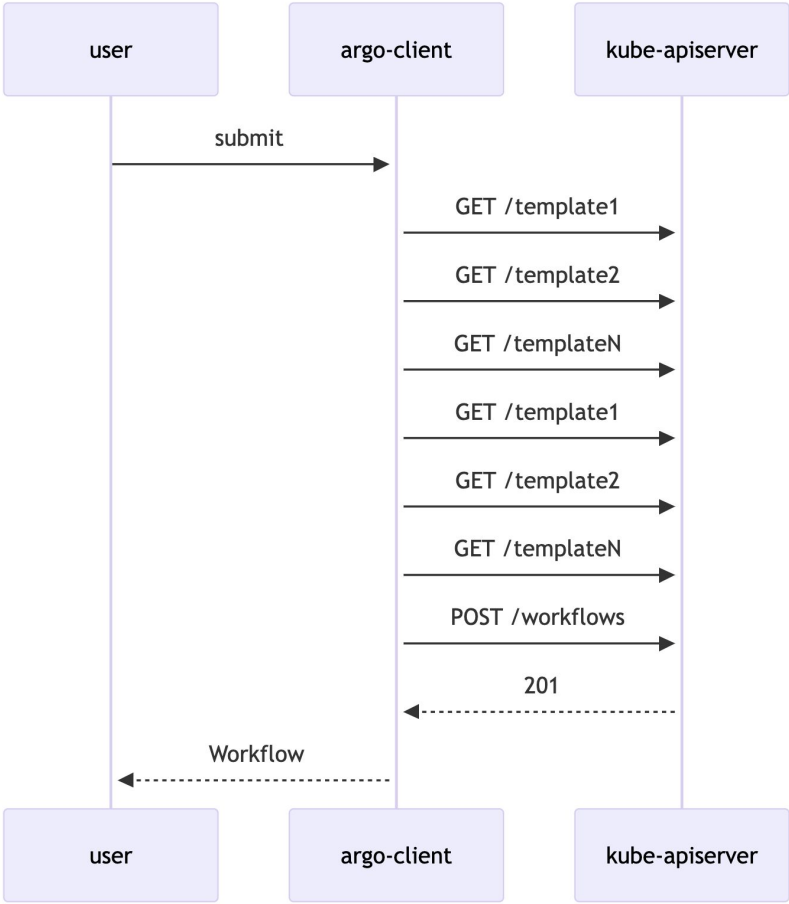
- template reference count
- parallel submissions count





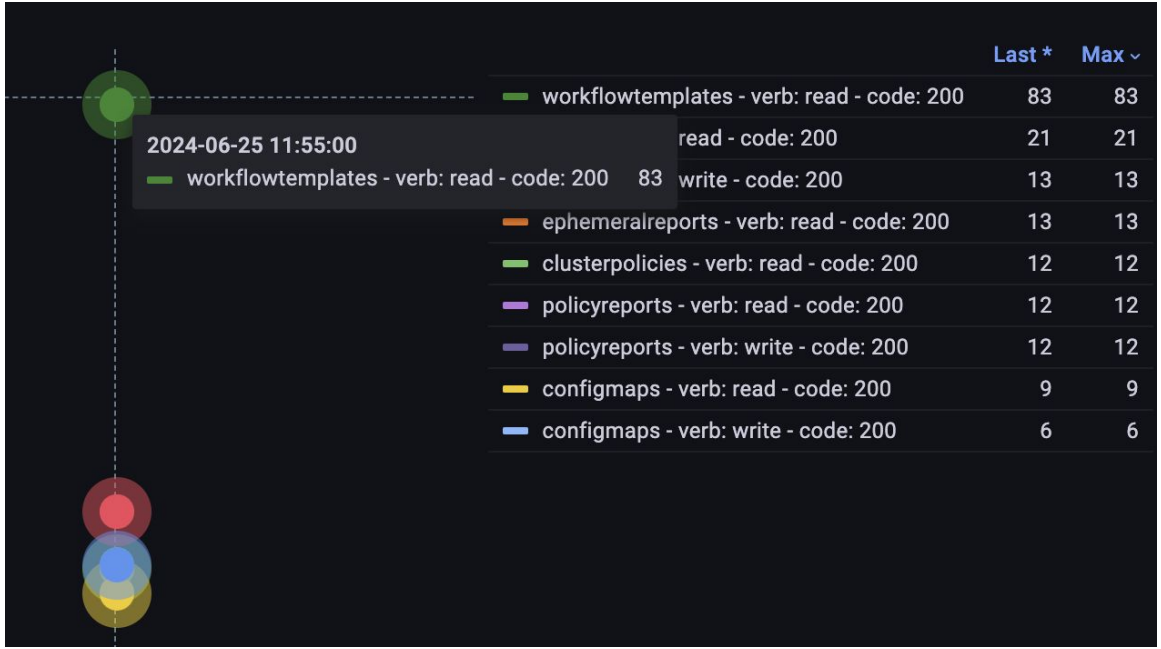
What really happens under the hood?

- Template validation
- Traversing over workflow document
- Resolution of template references



What's the damage ?

- Unbounded latency on submission
- High load on kube-apiserver / etcd
- Risk of hitting k8s api rate limits
- Resource utilization
  - network calls
  - request / response marshalling
  - memory allocation
  - GC pressure




```
I0926 14:13:02.171876   82005 request.go:665] Waited for 1.167453792s due to client-side throttling, not
priority and fairness, request: GET:https://
```

Quick fixes (hacks / workarounds)

K8S API Client Side Rate Limiting

The K8S client library rate limits the messages that can go out.

If you frequently see messages similar to this in the Controller log (issued by the library):

```
Waited for 7.090296384s due to client-side throttling, not priority and fairness, request: GET
```

Or, in >= v3.5, if you see warnings similar to this (could be any CR, not just `WorkflowTemplate`):

```
Waited for 7.090296384s, request:GET:https://10.100.0.1:443/apis/argoproj.io/v1alpha1/namespac
```

Then, if your K8S API Server can handle more requests:

- Increase both `--qps` and `--burst` arguments for the Controller. The `qps` value indicates the average number of queries per second allowed by the K8S Client. The `burst` value is the number of queries/sec the Client receives before it starts enforcing `qps`, so typically `burst` > `qps`. If not set, the default values are `qps=20` and `burst=30` (as of v3.5 (refer to `cmd/workflow-controller/main.go` in case the values change)).

(*) same principle applies to all components that call kube-apiserver (Argo Server/Argo CLI/SDK)

Quick fixes (hacks / workarounds)

Argo server’s k8s client rate limit override

```
62  +       extraArgs+: [
63  +           "--kube-api-burst=3000",
64  +           "--kube-api-qps=2000"
65  +       ],
```

Quick fixes (hacks / workarounds)

Argo SDK k8s client rate limit override

```
        ClientConfigSupplier: func() clientcmd.ClientConfig {
            return &customizedRestClientConfig{
                qps: client.qps,
            }
        },
func (c *customizedRestClientConfig) ClientConfig() (*restclient.Config, error) {
    restCfg, _ := c.getDeferredConfig().ClientConfig()
    restCfg.QPS = float32(c.qps)
    restCfg.Burst = 2 * c.qps
    return restCfg, nil
}
```

How to deal with it

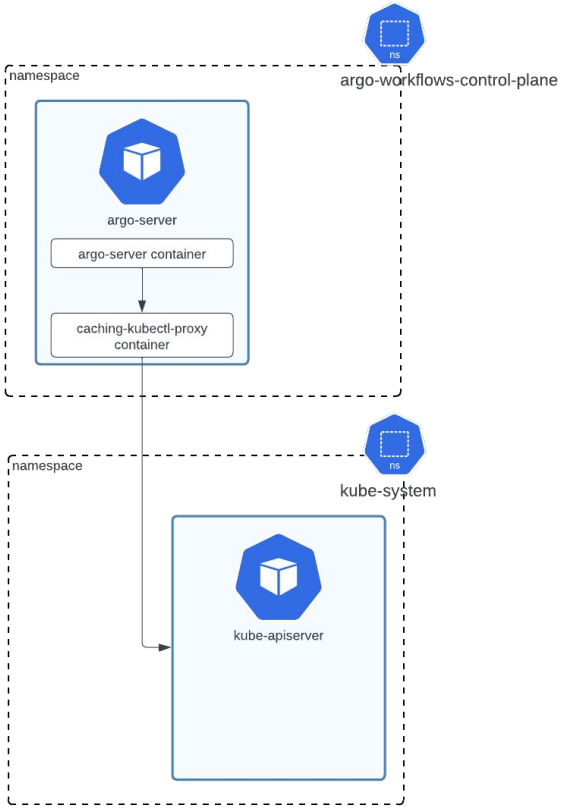
Post Workflow CR via k8s client and hope for the best

```
[➜  ~ time kubectl create -f echo-demo.yml
workflow.argoproj.io/320-echos created
kubectl create -f echo-demo.yml  0.31s user 0.09s system 12% cpu 3.288 total
[➜  ~ echo $?
0   resource creation succeeded
[➜  ~ kubectl get workflow 320-echos
NAME         STATUS    AGE
320-echos    Failed    19s  controller failed to process malformed resource
➜  ~
```

How to deal with it

Cache kube-apiserver responses @ Argo Server end

```go
func NewServer(filebase string, apiProxyPrefix string, staticPrefix string, filter *FilterServer, cfg
    memcached, err := memory.NewAdapter(
        memory.AdapterWithAlgorithm(memory.LRU),
        memory.AdapterWithCapacity(10000000),
    )
    if err != nil {
        log.Fatal(err)
    }
    cacheClient, err := cache.NewClient(
        cache.ClientWithAdapter(memcached),
        cache.ClientWithTTL(10*time.Minute),
        cache.ClientWithRefreshKey("opn"),
    )
    if err != nil {
        log.Fatal(err)
    }
    proxyHandler, err := NewProxyHandler(apiProxyPrefix, filter, cfg, keepalive, appendLocationPath)
    if err != nil {
        return nil, err
    }
    mux := http.NewServeMux()
    mux.Handle(apiProxyPrefix, cacheClient.Middleware(proxyHandler))
    if filebase != "" {
        // Require user to explicitly request this behavior rather than
        // serving their working directory by default.
        mux.Handle(staticPrefix, newFileHandler(staticPrefix, filebase))
    }
    return &Server{handler: mux}, nil
}
```



How to deal with it

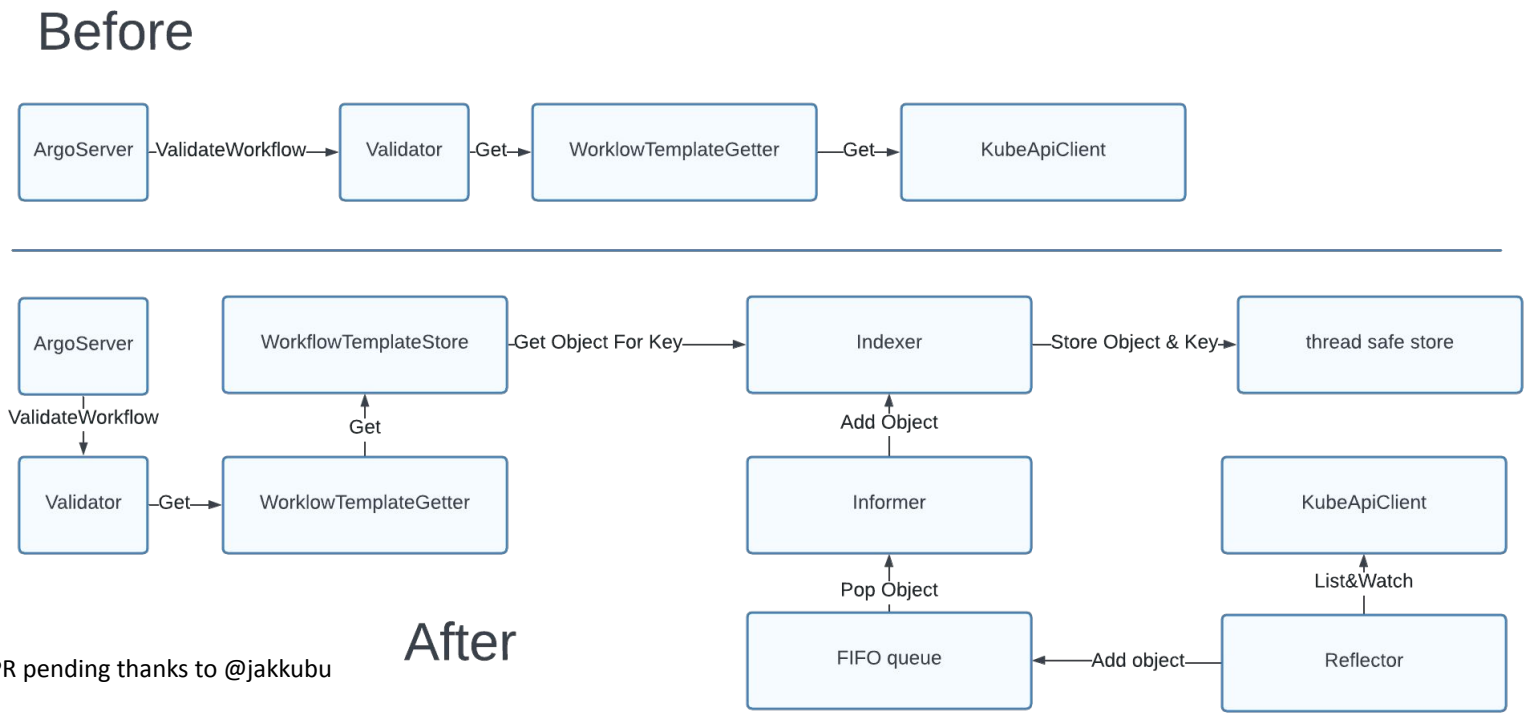
Cache kube-apiserver responses @ Argo client end

```go
func (c *customizedRestClientConfig) ClientConfig() (*restclient.Config, error) {
    restCfg, err := c.getDeferredConfig().ClientConfig()
    log := logging.Global()
    if err != nil {
        return nil, err
    }
    restCfg.Wrap(func(rt http.RoundTripper) http.RoundTripper {
        newRT, err := NewCache(rt)
        if err != nil {
            log.Error(err, msg: "Cannot start caching backend")
            return rt
        }
        return newRT
    })

    restCfg.QPS = float32(c.qps)
    restCfg.Burst = 2 * c.qps
    return restCfg, nil
}
```

```go
func NewCache(rt http.RoundTripper) (*TemplateCache, error) {
    cfg := bigcache.DefaultConfig(1 * time.Minute)
    cfg.HardMaxCacheSize = 512
    cfg.Shards = 8
    cfg.MaxEntriesInWindow = 100
    cfg.MaxEntrySize = 1e5
    cfg.Logger = newCacheLogger()
    cache, err := bigcache.New(context.Background(), cfg)
    if err != nil { return nil, fmt.Errorf( format: "error while

    return &TemplateCache{
        rt:    rt,
        cache: cache,
        re:    regexp.MustCompile(tmplRegex),
    }, nil
}
```

How to deal with it

Fix the root cause of a problem (add template caching support in workflow validation function)



(*) Upstream PR pending thanks to @jakkubu

Rate & Reach out

ArgoCon
NORTH AMERICA

https://www.linkedin.com/in/grzegorzpomykala/

https://sched.co/1izrZ