CLOUD NATIVE & KUBERNETES
AI DAY
NORTH AMERICA

SkyRay: Seamlessly Extending KubeRay to Multi-Cluster Multi-Cloud Operation

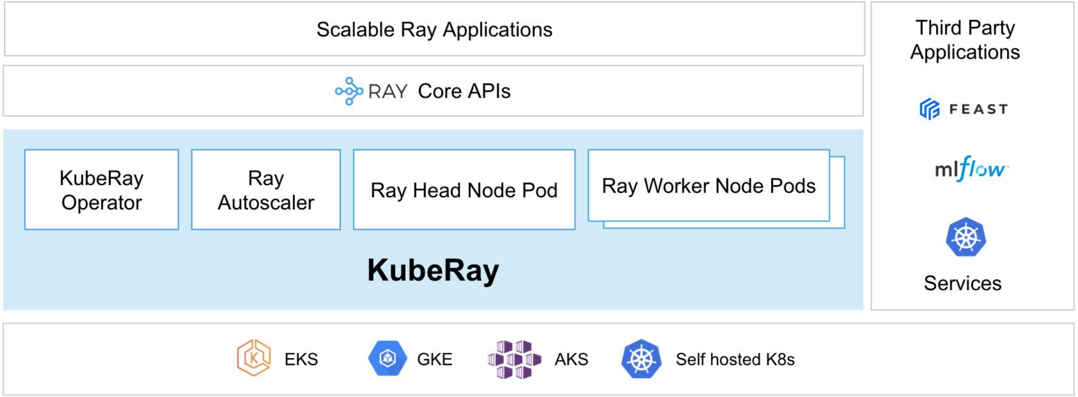
Anne Holler, Chief Scientist, Elotl

Cloud Native & Kubernetes AI Day, 11/12/24

SkyRay Overview: Ray and KubeRay Overview

Ray: unified framework that can scale ML/AI applications from laptop to cluster

KubeRay: creation, deletion, scaling of Ray clusters/jobs/services on K8s cluster



KubeRay Structure

SkyRay Overview: Multi-cluster Multi-cloud K8s Sky

Reasons to use **multiple K8s clusters** to group resources by characteristics include:

- **Service continuity**, e.g., limit impact of region outage
- **Workload purpose**, e.g., production (higher QoS) vs development (lower QoS)
- **Resource availability and cost**, e.g., per region GPU availability, per region pricing
- **Geo-location**, e.g., to satisfy workload location restrictions or to reduce response latency
- **Usage lifetime**, e.g., ML training or RAG ingestion may be periodic, ML serving is ongoing

Reasons to use K8s clusters from **multiple cloud vendors** include:

- **Avoids cloud vendor lock-in**
- **Supports customer cloud choice**
- **Allows differentiated costs**, e.g., fixed (on-premise, reserved) and dynamic (on-demand or spot)

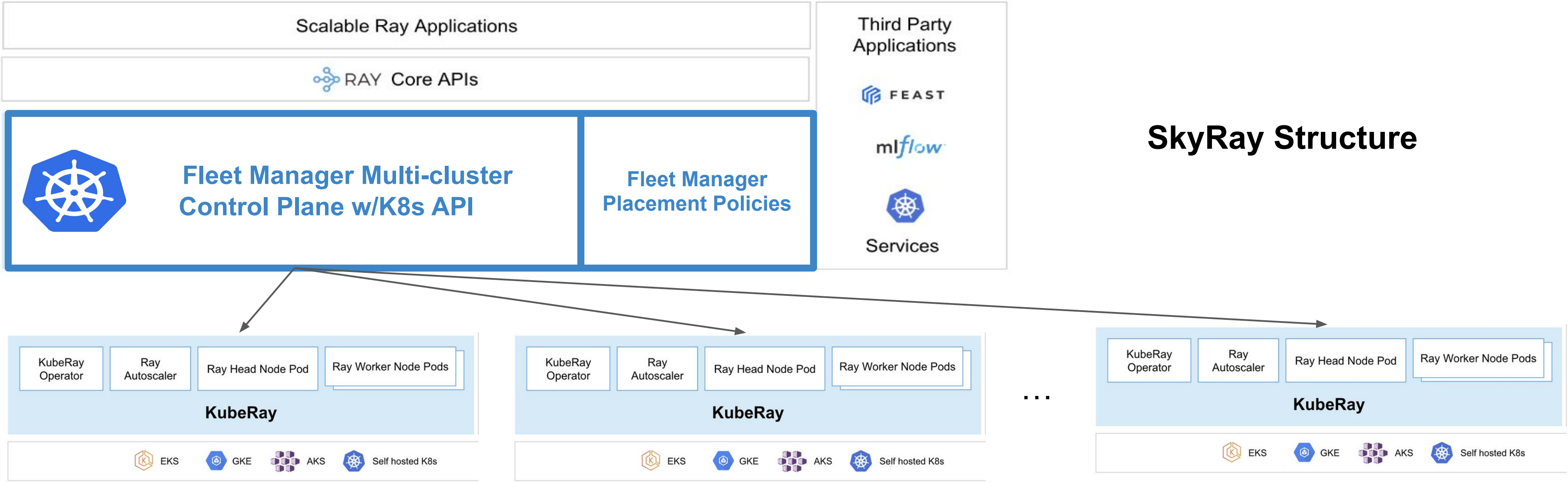
From Cloud Computing To Sky Computing [HotOS21]: need **commodity cloud compute layer**

- **K8s can represent a form of commodity compute layer for multi-cluster and multi-cloud use**
  - Requires using multi-cluster & multi-cloud K8s be nearly as **simple** as using single K8s cluster

SkyRay Overview: SkyRay Vision

SkyRay seamlessly extends KubeRay from K8s cluster to multi-cluster multi-cloud operation

- **Achieved via policy-driven K8s fleet mgr (FM), presenting K8s API, interoperating w/KubeRay**
  - FM deploys KubeRay on each of a set of workload K8s clusters, as per FM policy
  - FM places KubeRay clusters/jobs/services on workload clusters, chosen as per FM policies
  - Some example K8s FMs include Karmada, Open Cluster Management, and Nova



SkyRay Structure

Talk Outline

SkyRay Overview

SkyRay Operation

SkyRay Example ML/AI Use Cases

Conclusion

KubeRay Operation

KubeRay: K8s operator that simplifies deployment and management of Ray apps on K8s.

KubeRay supports three key Custom Resource Definitions:

- **RayCluster**
    - For creating a Ray cluster with the specified resources and attributes.
- **RayJob**
    - For creating a Ray cluster and submitting a job to it when the cluster is ready.
    - Can optionally delete the Ray cluster once the job finishes.
    - **Often used for ML/AI training.**
- **RayService**
    - For creating a Ray cluster and running a Ray Serve deployment graph.
    - Offers zero-downtime upgrades for Ray cluster, high availability, and Ray Serve autoscaling.
    - **Often used for ML/AI serving.**

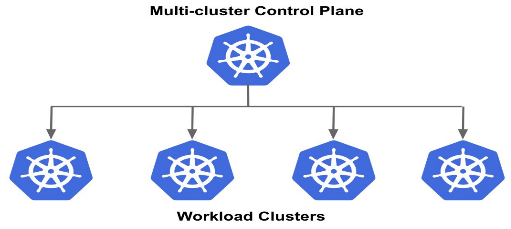
KubeRay deployments can optionally include the **Ray Autoscaler** for RayCluster scaling

- Automatically adds and removes workers from a RayCluster based on resource requests

SkyRay Operation

SkyRay operates as follows:

- **Fleet Manager K8s control plane tracks K8s workload clusters' characteristics**
  - E.g., name, cloud provider, region, available capacity, labels, K8s version, cluster autoscaler, …
- **Fleet Manager schedules KubeRay and its CRDs on all K8s workload clusters**
  - Specified by spread/duplicate Fleet Manager policy
- **All KubeRay cluster/job/service CR placement requests submitted to Fleet Manager K8s endpoint**
  - FM chooses workload cluster for requests based on cluster characteristics and placement policies



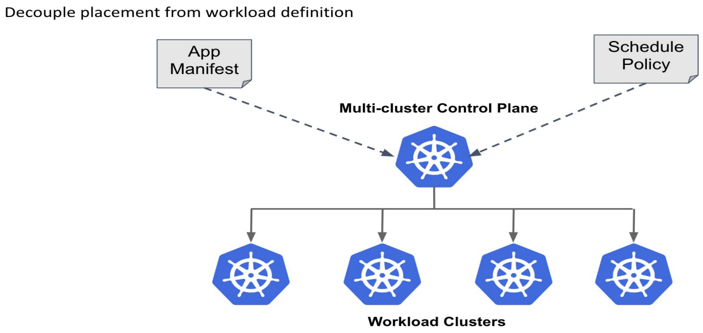
- Deploy workloads to one or more clusters from a central scheduler
- Aggregate view of workload topologies
- Orchestrate actions across workloads

Karmada, Admiralty, Elotl Nova follow similar architecture.

SkyRay K8s Fleet Manager Operation

For our SkyRay work, we used Elotl Nova K8s Fleet Manager, includes relevant capabilities:

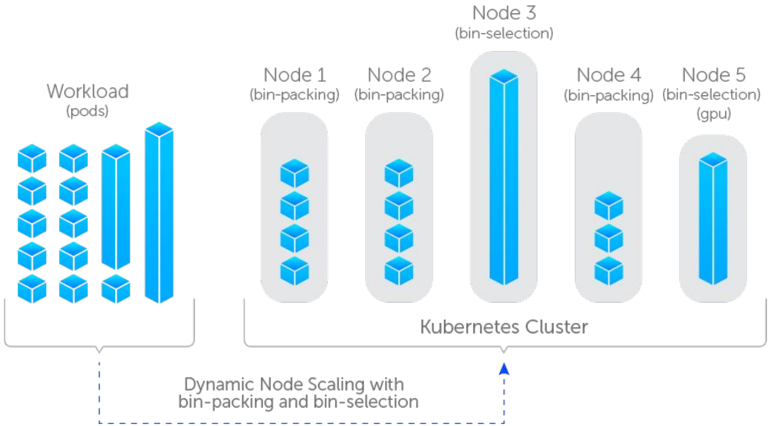
- **Policies: spread w/duplicate** (or percent), **specified cluster**, **priority, available-capacity**
  - Nova workloads placed using an **available-capacity policy are gang-scheduled**, meaning no part of a workload is scheduled until all of its components can be placed.
  - **ML/AI training jobs typically require gang-scheduling,** to allow all components of the distributed training task to make coordinated progress
- **Features: Just-in-time clusters [scale-to-0 or delete when idle, clone cluster]**, Overrides
  - JIT clusters can reduce costs or increase capacity for certain SkyRay scenarios


Decouple placement from workload definition
App Manifest
Schedule Policy
Multi-cluster Control Plane
Workload Clusters

Nova Fleet Manager Autoscaler-Aware Operation

When multiple clusters satisfy a workload's placement policy, Nova

- **Selects target cluster with existing sufficient available cluster resources, if any exists,**
- **Else selects target cluster with cluster autoscaler**, either K8s Cluster Autoscaler or Luna
  - **For our SkyRay work, we used the Luna smart cluster autoscaler**
    - Nova adds Luna's default pod placement label to workloads it schedules
  - **ML/AI workloads often require expensive GPU resources**
    - **Scaling GPU while needed from least expensive satisfactory type can reduce costs**



Luna Operation

Talk Outline

SkyRay Overview

SkyRay Operation

SkyRay Example ML/AI Use Cases

Conclusion

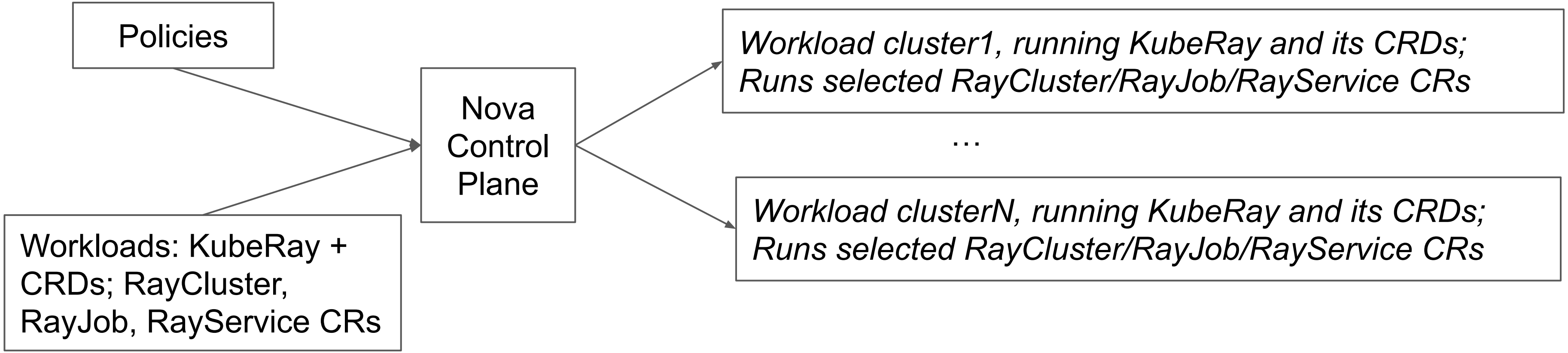
SkyRay Examples Overview

SkyRay recap

- **Nova CP uses spread/duplicate policy to schedule KubeRay & CRDs to all workload clusters**
- **Nova CP schedules each RayCluster/RayJob/RayService CR to workload cluster**
- **KubeRay on workload cluster materializes head and worker pods for CR as usual**

Scripts and K8s yaml for the examples available in github repo elotl/skyray

- **Some examples include dynamic allocation via Nova JIT or via Ray + Luna autoscaling**
- **One includes Nova CP scheduling KubeRay + non-KubeRay objects for Compound AI**


Policies
Workloads: KubeRay + CRDs; RayCluster, RayJob, RayService CRs
Nova Control Plane
Workload cluster1, running KubeRay and its CRDs; Runs selected RayCluster/RayJob/RayService CRs
…
Workload clusterN, running KubeRay and its CRDs; Runs selected RayCluster/RayJob/RayService CRs

SkyRay Example ML/AI Use Cases

RayJobs Training production ML/AI models
RayJobs Training experimental ML/AI models
RayServices Serving production vs development ML/AI models
RayServices Serving LLM models multi-cloud by priority with suspend/resume JIT
RayService Serving LLM model, no downtime K8s upgrade using delete/create JIT
CompoundAI RAG+LLM Serving with RayService for LLM

Image from KubeRay blog at AnyScale website

Initial Setup for Scenario Examples

For simplicity of presentation, all scenario examples involve 2 workload clusters

Initial view from the Nova control Plane, after KubeRay spread/duplicate scheduled

```
$ kubectl get all --all-namespaces
NAMESPACE   NAME                           TYPE        CLUSTER-IP     EXTERNAL-IP    PORT(S)     AGE
default     service/kuberay-operator       ClusterIP   10.96.241.6    <none>         8080/TCP    91s
default     service/kubernetes             ClusterIP   10.96.0.1      <none>         443/TCP     6m50s

NAMESPACE   NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
kuberayns   deployment.apps/kuberay-operator       2/1     2            2           91s
```

Scenario examples expect SKYRAY_PATH env var set to clone of github repo elotl/skyray

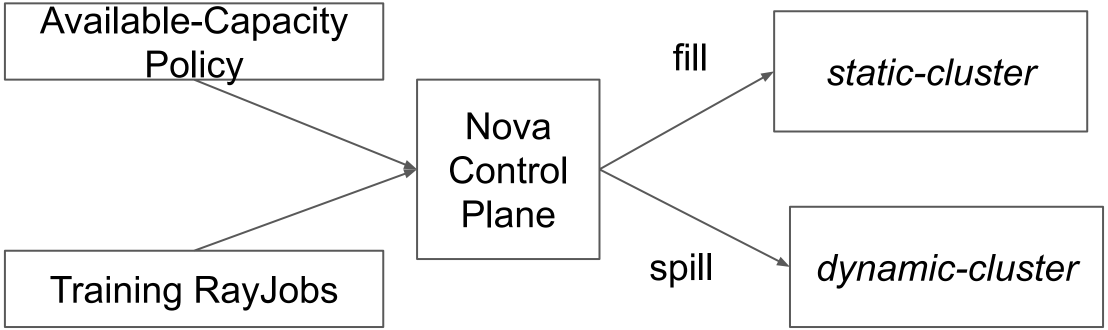
Scenario: RayJobs Training Production Models

For RayJobs training production ML/AI models on GPUs, desired behavior is "**fill and spill**"

- **Gang-scheduled on statically-allocated cluster if fit, else cluster w/dynamically-allocated resources**
- **Workloads' high value warrants on-demand resources; training so latency to scale resources ok**

Nova available-capacity placement policy set to match both clusters; Nova gang-schedules

- Nova places training RayJobs on *static-cluster* preferentially since resources immediately available.
- When a training RayJob arrives that does not fully fit on *static-cluster*
  - Nova places it on *dynamic-cluster* and Luna adds resources to accommodate it



On premise or reserved cloud resources
No cluster autoscaler running

On-demand cloud resources
w/a few static nodes running KubeRay
& Luna cluster autoscaler

RayJobs Training Production Models Example Overview

As a proxy for a production training workload, we use Pytorch image train benchmark

- **Run as a RayJob deployed using KubeRay, adapted from the example here.**

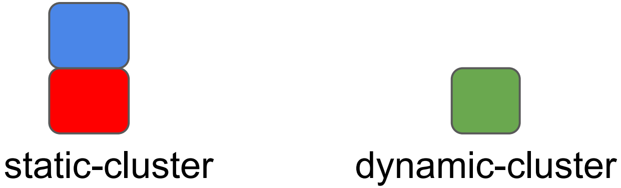
PyTorch

Each RayJob's RayCluster is configured with CPU head and 2 single-GPU workers.

- The configuration of the RayJob with its associated RayCluster is available here.
- The RayJob available-capacity placement policy is here.

Three copies of the RayJob are deployed to the Nova Control plane.

- **Nova gang-schedules first 2 copies on static-cluster, since they fit there**
- **Nova gang-schedules third copy on dynamic-cluster, Luna scales up cluster resources**
- Experiment run with nova control plane and workload clusters on EKS



RayJobs Training Production Models Example

Deploy 1st training job instance in rayjob1 namespace, scheduled on *static-cluster*

```
$ export RAYCLUSTER_NAMESPACE1=rayjob1
$ ${SKYRAY_PATH}/deploy-scripts/deploy-rayjob-train.sh ${SKYRAY_PATH} ${RAYCLUSTER_NAMESPACE1} ${AWS_ACCESS_KEY_ID}
${AWS_SECRET_ACCESS_KEY}
<snip>
$ export TARG_CLUSTER1=$(kubectl get rayjob.ray.io/rayjob-train -n ${RAYCLUSTER_NAMESPACE1} -L nova.elotl.co/target-cluster |
awk {'print $NF'} | tail -1)
$ echo ${TARG_CLUSTER1}
static-cluster
```

Deploy 2nd training job instance in rayjob2 namespace, also scheduled on *static-cluster*

```
$ export RAYCLUSTER_NAMESPACE2=rayjob2
$ ${SKYRAY_PATH}/deploy-scripts/deploy-rayjob-train.sh ${SKYRAY_PATH} ${RAYCLUSTER_NAMESPACE2} ${AWS_ACCESS_KEY_ID}
${AWS_SECRET_ACCESS_KEY}
<snip>
$ export TARG_CLUSTER2=$(kubectl get rayjob.ray.io/rayjob-train -n ${RAYCLUSTER_NAMESPACE2} -L nova.elotl.co/target-cluster |
awk {'print $NF'} | tail -1)
$ echo ${TARG_CLUSTER2}
static-cluster
```

Deploy 3rd training job instance in rayjob3 namespace, scheduled on *dynamic-cluster*

```
$ export RAYCLUSTER_NAMESPACE3=rayjob3
$ ${SKYRAY_PATH}/deploy-scripts/deploy-rayjob-train.sh ${SKYRAY_PATH} ${RAYCLUSTER_NAMESPACE3} ${AWS_ACCESS_KEY_ID}
${AWS_SECRET_ACCESS_KEY}
<snip>
$ export TARG_CLUSTER3=$(kubectl get rayjob.ray.io/rayjob-train -n ${RAYCLUSTER_NAMESPACE3} -L nova.elotl.co/target-cluster |
awk {'print $NF'} | tail -1)
$ echo ${TARG_CLUSTER3}
dynamic-cluster
```

“Fill and spill” achieved via available-capacity policy

Multi-cluster deployment as simple as single cluster deployment; view jobs from Nova CP

```
$ kubectl get all --all-namespaces <SNIP>
NAMESPACE NAME                         JOB STATUS   DEPLOYMENT STATUS   START TIME              END TIME    AGE
rayjob1   rayjob.ray.io/rayjob-train    RUNNING     Running             2024-07-01T22:13:02Z                9m11s
rayjob2   rayjob.ray.io/rayjob-train    RUNNING     Running             2024-07-01T22:12:07Z                4m55s
rayjob3   rayjob.ray.io/rayjob-train                Initializing        2024-07-01T22:16:28Z                34s
```

Luna scales up dynamic cluster accordingly

```
$ kubectl --context=dynamic-cluster get nodes -Lnode.kubernetes.io/instance-type
NAME                                          STATUS   ROLES    AGE     VERSION               INSTANCE-TYPE
ip-192-168-161-254.us-west-2.compute.internal Ready    <none>   4m47s   v1.29.3-eks-ae9a62a   t3a.2xlarge
ip-192-168-182-75.us-west-2.compute.internal  Ready    <none>   55m     v1.29.3-eks-ae9a62a   t3a.small
ip-192-168-61-229.us-west-2.compute.internal  Ready    <none>   4m24s   v1.29.3-eks-ae9a62a   g4dn.2xlarge
ip-192-168-63-192.us-west-2.compute.internal  Ready    <none>   4m27s   v1.29.3-eks-ae9a62a   g4dn.2xlarge
ip-192-168-94-42.us-west-2.compute.internal   Ready    <none>   56d     v1.29.3-eks-ae9a62a   m5.large
```

All 3 RayJobs run to completion

```
$ kubectl get all --all-namespaces <SNIP>
NAMESPACE NAME                       JOB STATUS DEPL STATUS START TIME           END TIME                AGE
rayjob1   rayjob.ray.io/rayjob-train SUCCEEDED  Complete   2024-07-01T22:13:02Z 2024-07-01T22:26:30Z    22m
rayjob2   rayjob.ray.io/rayjob-train SUCCEEDED  Complete   2024-07-01T22:12:07Z 2024-07-01T22:19:49Z    18m
rayjob3   rayjob.ray.io/rayjob-train SUCCEEDED  Complete   2024-07-01T22:16:28Z 2024-07-01T22:30:27Z    14m
```

Luna scales down the dynamic cluster, after rayjob3 removed

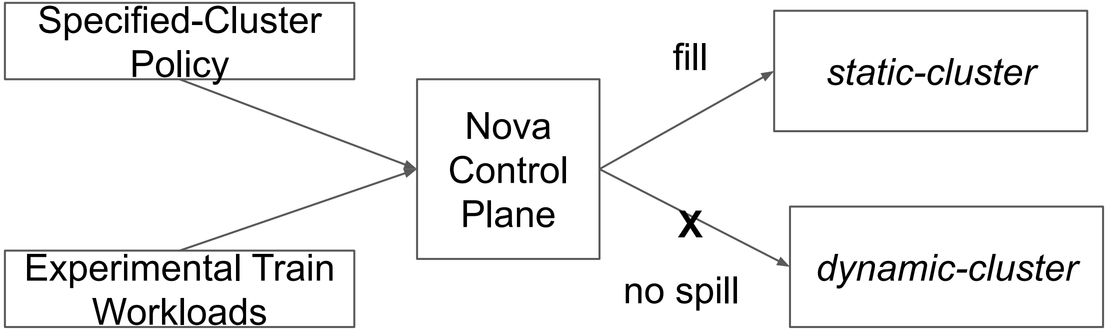
Scenario: RayJobs Training Experimental Models

For training experimental ML/AI models on GPUs, desired behavior is "**fill, no spill**"

- **Workloads scheduled on statically-allocated on-premise or reserved cluster, sunk-cost**
- **Workloads have not proven value to warrant paying for on-demand cloud resources**

Nova specified-cluster placement policy is set to match only static-cluster

- Nova places all experimental training workloads on the cluster
- Any workloads that cannot be admitted are pending in the cluster.



On premise or reserved cloud resources
No cluster autoscaler running

On-demand cloud resources
w/a few static nodes running KubeRay
& Luna cluster autoscaler

RayJobs Training Experimental Models Example Overview

As proxy for experimental training workload, we again use Pytorch image train benchmark

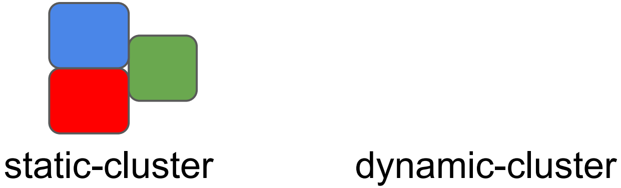
- **Run as a RayJob deployed using KubeRay, adapted from the example here**

Each RayJob's RayCluster is again configured with a CPU head and 2 single-GPU workers.

- The configuration of the RayJob with its associated RayCluster is available here.
- The RayJob cluster-specific placement policy is here.

Three copies of the RayJob are deployed to the Nova Control plane.

- **Nova schedules all 3 on static-cluster**
- **Objects that do have sufficient resources to schedule stay pending**



RayJobs Training Experimental Models

Deploy 3 training job instances in separate namespaces, all scheduled on *static-cluster*

```
$ export RAYCLUSTER_NAMESPACE=rayjob1
$ ${SKYRAY_PATH}/deploy-scripts/deploy-rayjob-train-static.sh ${SKYRAY_PATH} ${RAYCLUSTER_NAMESPACE} ${AWS_ACCESS_KEY_ID}
${AWS_SECRET_ACCESS_KEY}
<Snip>
$ export TARG_CLUSTER=$(kubectl get rayjob.ray.io/rayjob-train -n ${RAYCLUSTER_NAMESPACE} -L nova.elotl.co/target-cluster | awk
{'print $NF'} | tail -1)
$ echo ${TARG_CLUSTER}
Static-cluster

$ export RAYCLUSTER_NAMESPACE=rayjob2
$ ${SKYRAY_PATH}/deploy-scripts/deploy-rayjob-train-static.sh ${SKYRAY_PATH} ${RAYCLUSTER_NAMESPACE} ${AWS_ACCESS_KEY_ID}
${AWS_SECRET_ACCESS_KEY}
<snip>
$ export TARG_CLUSTER=$(kubectl get rayjob.ray.io/rayjob-train -n ${RAYCLUSTER_NAMESPACE} -L nova.elotl.co/target-cluster | awk
{'print $NF'} | tail -1)
$ echo ${TARG_CLUSTER}
static-cluster

$ export RAYCLUSTER_NAMESPACE=rayjob3
$ ${SKYRAY_PATH}/deploy-scripts/deploy-rayjob-train-static.sh ${SKYRAY_PATH} ${RAYCLUSTER_NAMESPACE} ${AWS_ACCESS_KEY_ID}
${AWS_SECRET_ACCESS_KEY}
<snip>
$ export TARG_CLUSTER=$(kubectl get rayjob.ray.io/rayjob-train -n ${RAYCLUSTER_NAMESPACE} -L nova.elotl.co/target-cluster | awk
{'print $NF'} | tail -1)
$ echo ${TARG_CLUSTER}
Static-cluster
```

“Fill, no spill” achieved via specified-cluster policy

Again, multi-cluster deployment as simple as single cluster deployment

In this case, *static-cluster* doesn’t have sufficient remaining resources to run 3rd RayJob.

- **Its unschedulable pods stay pending until capacity freed up by removal of previous job(s).**

```
$ kubectl get all --all-namespaces
. . .
NAMESPACE NAME                          JOB STATUS DEPL STATUS START TIME             END TIME               AGE
rayjob1   rayjob.ray.io/rayjob-train SUCCEEDED Complete     2024-07-02T13:49:21Z   2024-07-02T13:56:49Z   8m5s
rayjob2   rayjob.ray.io/rayjob-train RUNNING   Running      2024-07-02T13:53:16Z                          4m10s
rayjob3   rayjob.ray.io/rayjob-train           Initializing 2024-07-02T13:54:47Z                          2m39s
…
$ kubectl get all --all-namespaces
. . .
NAMESPACE NAME                          JOB STATUS DEPL STATUS START TIME             END TIME               AGE
rayjob2    rayjob.ray.io/rayjob-train SUCCEEDED  Complete    2024-07-02T13:53:16Z   2024-07-02T14:00:49Z   12m
rayjob3    rayjob.ray.io/rayjob-train RUNNING    Running     2024-07-02T13:54:47Z                          11m
…
$ kubectl get all --all-namespaces
. . .
NAMESPACE NAME                          JOB STATUS DEPL STATUS   START TIME             END TIME               AGE
rayjob2    rayjob.ray.io/rayjob-train SUCCEEDED Complete       2024-07-02T13:53:16Z   2024-07-02T14:00:49Z   14m
rayjob3    rayjob.ray.io/rayjob-train SUCCEEDED Complete       2024-07-02T13:54:47Z   2024-07-02T14:07:53Z   13m
```

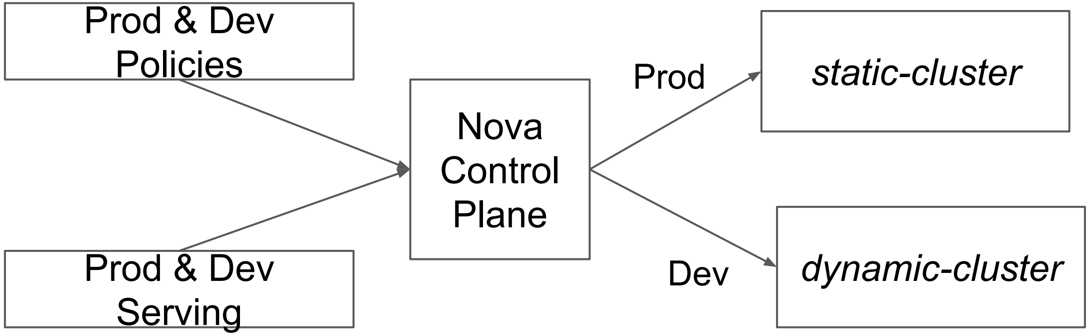
Scenario: RayServices Serving Prod vs Dev AI/ML Models

For RayServices serving prod vs dev models, desired behavior is "**select the right cluster**"

- **Online prod serving placed on statically-allocated cluster configured for max load SLA**
    - **Low latency required**, serving on critical path of business app (e.g., predicting ride-share ETA)
- **Dev serving placed on cluster w/dynamically-allocated lower-cost resources**
    - **Low latency not required**

Nova cluster selection policies ([prod](), [dev]()) set to match cluster label for workload class

- **Label *production* on *static-cluster* cluster CR, *development* on *dynamic-cluster* cluster CR**
    - **Note: Cluster labels add layer of indirection that facilitates changing set of category clusters**



On premise or reserved cloud resources
No cluster autoscaler running
A10G GPU instances statically allocated

On-demand cloud resources
w/a few static nodes running KubeRay
and Luna cluster autoscaler
T4 GPU instances dynamically allocated

RayServices Serving Prod vs Dev Models Example Overview

As a representative serving workload, we use the **text summarizer model service**

- Run as a **RayService** deployed using KubeRay, adapted from the example here.

Each RayService's RayCluster is configured with a CPU head and 1 single-GPU worker

- The configuration of the RayService with its associated RayCluster is available here
- These are the RayService cluster selection policies for production, development

Two copies of the RayService are deployed to the Nova Control plane.

- **The RayService deployed to the production namespace is scheduled on *static-cluster***
- **The RayService deployed to the development namespace is scheduled on *dynamic-cluster***

🟩 static-cluster

🟥 dynamic-cluster

RayService Production Serving

Production namespace deployed to Nova CP; Placed w/ spread/duplicate policy

```
<SNIP>
```

RayService deployed to Nova CP in production namespace; Placed on *static-cluster*

```
$ kubectl apply -f ${SKYRAY_PATH}/deploy-scripts/ray-service.text-summarizer.yaml --namespace=production
rayservice.ray.io/text-summarizer created

$ kubectl --context=static-cluster get all -n production
NAME                                                            READY   STATUS    RESTARTS   AGE
pod/text-summarizer-raycluster-ntcfh-head-tmnqr                 1/1     Running   0          68m
pod/text-summarizer-raycluster-ntcfh-worker-gpu-group-wft6f     1/1     Running   0          68m
NAME                                                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                                        AGE
service/text-summarizer-head-svc                        ClusterIP   10.100.6.157     <none>        10001/TCP,8265/TCP,6379/TCP,8080/TCP,8000/TCP   60m
service/text-summarizer-raycluster-ntcfh-head-svc       ClusterIP   10.100.197.135   <none>        10001/TCP,8265/TCP,6379/TCP,8080/TCP,8000/TCP   68m
service/text-summarizer-serve-svc                       ClusterIP   10.100.205.162   <none>        8000/TCP                                        60m
NAME                                                     DESIRED WORKERS   AVAILABLE WORKERS   CPUS   MEMORY   GPUS   STATUS   AGE
raycluster.ray.io/text-summarizer-raycluster-ntcfh      1                 1                   5      20G      1      ready    68m
NAME                                  AGE
rayservice.ray.io/text-summarizer    68m
```

Production Serving Validated:

```
$ kubectl --context=static-cluster port-forward svc/text-summarizer-serve-svc 8000 -n production
<SNIP>
$ python text_summarizer_req.py
Paris is the capital and most populous city of France. It has an estimated population of 2,175,601 residents as of 2018. The City of Paris is the
centre of the French capital.
```

RayService Development Serving

Development namespace deployed to Nova control plane; Placed w/ spread/duplicate policy

```
<SNIP>
```

RayService deployed to Nova CP in development namespace; Placed on *dynamic-cluster*

```
$ kubectl apply -f ${SKYRAY_PATH}/deploy-scripts/ray-service.text-summarizer.yaml --namespace=development
rayservice.ray.io/text-summarizer created

$ kubectl --context=dynamic-cluster get all -n development
NAME                                                          READY   STATUS    RESTARTS   AGE
pod/text-summarizer-raycluster-2xnts-head-68bvm               1/1     Running   0          47m
pod/text-summarizer-raycluster-2xnts-worker-gpu-group-s8pbn   1/1     Running   0          47m

NAME                                               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                                         AGE
service/text-summarizer-head-svc                   ClusterIP   10.100.45.127   <none>        10001/TCP,8265/TCP,6379/TCP,8080/TCP,8000/TCP   37m
service/text-summarizer-raycluster-2xnts-head-svc   ClusterIP   10.100.46.227   <none>        10001/TCP,8265/TCP,6379/TCP,8080/TCP,8000/TCP   47m
service/text-summarizer-serve-svc                  ClusterIP   10.100.209.7    <none>        8000/TCP                                        37m

NAME                                                 DESIRED WORKERS   AVAILABLE WORKERS   CPUS   MEMORY   GPUS   STATUS   AGE
raycluster.ray.io/text-summarizer-raycluster-2xnts   1                 1                   5      20G      1      ready    47m

NAME                                AGE
rayservice.ray.io/text-summarizer   47m
```

Development Serving Validated

```
$ kubectl --context=dynamic-cluster port-forward svc/text-summarizer-serve-svc 8000 -n development
<SNIP>
$ python text_summarizer_req.py
Paris is the capital and most populous city of France. It has an estimated population of 2,175,601 residents as of 2018. The City of Paris is the
centre of the French capital.
```

“Select right cluster” achieved via label cluster-select policy

Again, multi-cluster deployment as simple as single cluster deployment

Prod on more costly statically-allocated GPU, dev on cheaper dynamically-allocated GPU

- ***static-cluster* was configured w/ g5.xlarge, which includes an NVIDIA A10G GPU**
- **Luna allocates a g4dn.xlarge for *dynamic-cluster*, which includes an NVIDIA T4 GPU**
    - us-east per-hr on-demand price for g4dn.xlarge < per-hr reserved price for g5.xlarge
    - g4dn.xlarge is good choice for dev workload, which doesn’t warrant more powerful GPU

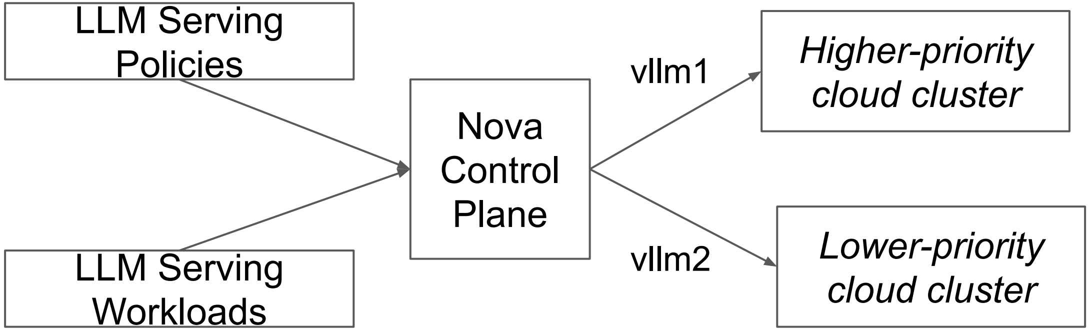
```
$ kubectl --context=dynamic-cluster get nodes -Lnode.kubernetes.io/instance-type
NAME                                            STATUS   ROLES    AGE   VERSION               INSTANCE-TYPE
ip-192-168-164-97.us-west-2.compute.internal    Ready    <none>   8d    v1.29.3-eks-ae9a62a   t3a.small
ip-192-168-171-101.us-west-2.compute.internal   Ready    <none>   48m   v1.29.3-eks-ae9a62a   t3a.xlarge
ip-192-168-49-24.us-west-2.compute.internal     Ready    <none>   48m   v1.29.3-eks-ae9a62a   g4dn.xlarge
ip-192-168-94-42.us-west-2.compute.internal     Ready    <none>   64d   v1.29.3-eks-ae9a62a   m5.large
```

Scenario: RayServices Serving LLM Models Multi-cloud Priority w/JIT

For serving LLM models w/ordered cloud preferences, desired "**select cluster by cloud priority**"

- **Nova cluster selection policy use cloud in priority order, set as per some criteria, e.g.**
  - **Higher-priority may be lower cost, lower-priority may have more resources**

- **JIT w/Suspend/Resume standby; cluster idle->nodes scaled to 0, restored when active**
  - **Option NOVA_NAMESPACES_EXCLUDE_ACTIVE set to Kuberay namespace *kuberayns*; So KubeRay running on workload clusters does not block cluster entering standby**



Multi-cloud Cloud Provider Clusters in Priority Examples

Presenting two cloud provider priority examples

- **E1: Fill/spill w/GKE & EKS clusters, shows on-demand w/alternative on-demand**
- **E2: Fill/spill w/Rackspace & EKS clusters, shows allocated spot w/alternative on-demand**

GKE cluster on GCE cloud provider; nodes:

- `2x e2-medium: 2 CPUs, 4 GB memory`
- `1x e2-standard-4: 4 CPUs, 16 GB memory`
- `1x g2-standard-16: 16 CPUs, 64 GB memory, 1 L4 GPU; node taint key: nvidia.com/gpu; effect: NoSchedule`

EKS cluster on AWS cloud provider; nodes:

- `2x m5.large: 2 CPUs, 8 GB memory`
- `1x t3a.xlarge: 4 CPUs, 16 GB memory`
- `1x g6.4xlarge: 16 CPUs, 64 GB memory, 1 L4 GPU; node taint key: nvidia.com/gpu; effect: NoSchedule`

Rackspace Spot cluster on Openstack cloud provider managed by Platform9; nodes:

- `1x` ao.2.24.128_A30 (24 CPUs, 128 GB memory, 1 A30 GPU)

E1: RayServices Serving LLM Models Multi-cloud Priority

EKS and GKE clusters initially idle and in standby, control plane costs $0.10/hr
KubeRay is pending with its ***kuberayns*** namespace set to not represent active use

```
$ kubectl get clusters
NAME                   K8S-VERSION   K8S-CLUSTER     REGION        ZONE            READY   IDLE   STANDBY
eks-workload-cluster   1.30          cpu-us-west-2   us-west-2     us-west-2d      False   True   True
gke-workload-cluster   1.29          anne-nova-cp    us-central1   us-central1-c   False   True   True
```

Spread/dupe vllm1 namespace for LLM deploy; Set cloud priority policy for workloads

```
$ export RAYCLUSTER_NAMESPACE=vllm1
$ envsubst < ${SKYRAY_PATH}/deploy-scripts/namespace.yaml | kubectl apply -f -
$ envsubst < ${SKYRAY_PATH}/policies/rayservicenspriorityp olicy.yaml | kubectl apply -f -
```

Deploy RayService to serve LLM model mosaicml/mpt-7b-chat using vLLM to vllm1 namespace; gke-workload-cluster exits standby and kuberay starts up rayservice head and worker on it

```
$ kubectl apply -f ${SKYRAY_PATH}/luna-llm-serve/ray-service.llm.yaml -n ${RAYCLUSTER_NAMESPACE}
$ kubectl wait --for=jsonpath='{.status.serviceStatus}'=Running rayservice.ray.io/llm-model-serve --namespace
${RAYCLUSTER_NAMESPACE} --timeout=20m
$ export TARG_CLUSTER=$(kubectl get rayservice.ray.io/llm-model-serve -n ${RAYCLUSTER_NAMESPACE} -L
nova.elotl.co/target-cluster | awk {'print $NF'} | tail -1)
$ echo ${TARG_CLUSTER}
gke-workload-cluster
```

“Select cluster by cloud priority” achieved via ordered-selector policy

Validate serving LLM model from vllm1 namespace via query.py

```
$ kubectl --context=${TARG_CLUSTER} port-forward service/llm-model-serve-serve-svc 8000:8000 -n ${RAYCLUSTER_NAMESPACE}
$ python query.py
Type your query here: what is a good christmas present for a 6 year old?
A good Christmas present for a 6-year-old could be a board game like Monopoly or a toy like a doll or a robot. Other ideas
could be books, puzzles, or arts and crafts supplies. It's important to consider the child's interests and hobbies when
choosing a gift.
```

Perform same ops for LLM model in namespace vllm2, workload spilled to awakened EKS cluster

```
$ export RAYCLUSTER_NAMESPACE=vllm2 <SNIP>
$ echo ${TARG_CLUSTER}
eks-workload-cluster
```

Validate serving LLM model from vllm2 namespace

```
…Type your query here: When is a good time to buy a stock?
As an AI language model, I don't have personal beliefs or opinions, but I can provide some general guidelines to help you make
informed decisions when buying stocks. 1. Research the company's financial health and performance. Look at its past earnings
reports, revenue growth, and debt-to-equity ratio. <SNIP>
```

Both workload clusters active; delete namespace to remove model, cluster re-enters standby

E2: RayServices Serving LLM Models Multi-cloud Priority

Rackspace & EKS clusters initially idle; KubeRay ***kuberayns*** namespace set to not be active use

```
$ kubectl get clusters
NAME                  K8S-VERSION   K8S-CLUSTER   REGION      ZONE         READY   IDLE   STANDBY
elotl-llmexperiment   1.29                        SJC3        nova         True    True   False
workload-cluster-1    1.30          gpu-cluster   us-west-2   us-west-2d   True    True   False
```

Spread/dupe vllm1 namespace for LLM deploy; Set cloud priority policy for workloads

```
$ export RAYCLUSTER_NAMESPACE=vllm1
$ envsubst < ${SKYRAY_PATH}/deploy-scripts/namespace.yaml | kubectl apply -f -
$ envsubst < ${SKYRAY_PATH}/policies/rayservicenspprioritypolicy.yaml | kubectl apply -f -
```

Deploy RayService to serve LLM model mosaicml/mpt-7b-chat using vLLM to vllm1 namespace; elotl-llmexperiment cluster chosen and kuberay starts up rayservice head and worker on it

```
$ kubectl apply -f ${SKYRAY_PATH}/luna-llm-serve/ray-service.llm.yaml -n ${RAYCLUSTER_NAMESPACE}
$ kubectl wait --for=jsonpath='{.status.serviceStatus}'=Running rayservice.ray.io/llm-model-serve --namespace
${RAYCLUSTER_NAMESPACE} --timeout=20m
$ export TARG_CLUSTER=$(kubectl get rayservice.ray.io/llm-model-serve -n ${RAYCLUSTER_NAMESPACE} -L nova.elotl.co/target-cluster
| awk {'print $NF'} | tail -1)
$ echo ${TARG_CLUSTER}
elotl-llmexperiment
```

“Select cluster by cloud priority” achieved via ordered-selector policy

Validate serving LLM model from vllm1 namespace via query.py

```
$ kubectl --context=${TARG_CLUSTER} port-forward service/llm-model-serve-serve-svc 8000:8000 -n ${RAYCLUSTER_NAMESPACE}
$ python query.py
Type your query here: what is the longest river in the world?
The longest river in the world is the Nile River, which flows through 11 countries in Africa. It has a length of 6,695 kilometers
(4,130 miles) and a drainage basin of 2,130,000 square kilometers (827,000 square miles).
```

Perform same ops for LLM model in namespace vllm2, workload spilled to EKS cluster

```
$ export RAYCLUSTER_NAMESPACE=vllm2 <SNIP>
$ echo ${TARG_CLUSTER}
eks-workload-cluster
```

Validate serving LLM model from vllm2 namespace

```
Type your query here: what is zumba?
Zumba is a fitness program that combines dance and aerobic exercise. It was created by Alberto "Beto" Perez in 2002 and has since become a
global phenomenon, with millions of people taking part in Zumba classes around the world. Zumba workouts are designed to be fun and
energetic, and they typically involve a combination of fast-paced dancing, aerobic exercise, and strength training. Zumba classes are led
by instructors who teach the moves and provide motivation and encouragement to participants.
```

Both workload clusters active; again, can delete namespace to remove model

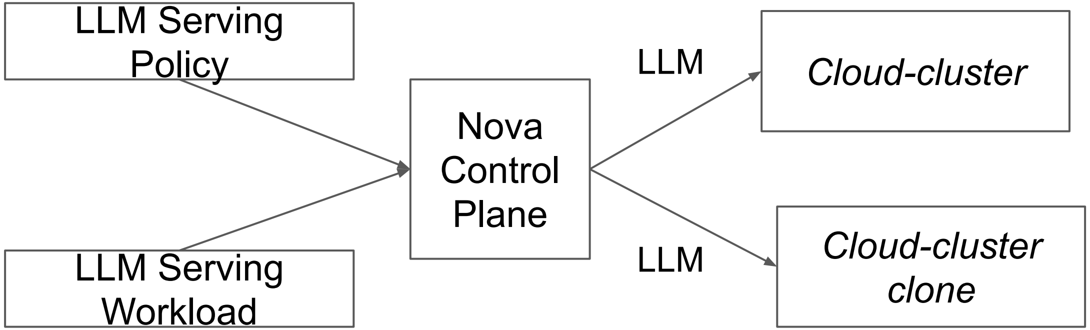
Scenario: LLM Model Serving w/No downtime K8s upgrade

For upgrade K8s cluster serving LLM model, desired behavior “**clone-upgrade/spread/validate**”

- **RayService LLM serving on K8s cluster w/ spread/duplicate on labeled Nova cluster policy**
- **Nova JIT w/Delete/Create standby enabled**

User adds clone of labeled Nova cluster CR w/updated K8s version, K8s and Nova names

- **Nova JIT creates corresponding cluster, Nova schedules dupe LLM serving workload on it**
- **User validates LLM serving workload on upgraded K8s cluster, chooses when to switch DNS**
  - User then removes label from old Nova cluster CR, cluster becomes idle and JIT deletes it
    User can then remove old cluster CR



LLM Model Serving w/No downtime K8s upgrade, Start

Label applied to workload cluster

```
$ kubectl label cluster workload-cluster-1 cluster.type=rayservice
$ kubectl get clusters --context=nova --show-labels
NAME                  K8S-VERSION   K8S-CLUSTER     REGION       ZONE          READY   IDLE   STANDBY   LABELS
workload-cluster-1    1.30          cpu-us-west-2   us-west-2    us-west-2d    True    True   False
cluster.type=rayservice,kubernetes.io/metadata.name=workload-cluster-1,nova.elotl.co/cluster.novacreated=false,nova.elotl.co/cl
uster.provider=aws,nova.elotl.co/cluster.region=us-west-2,nova.elotl.co/cluster.version=1.30,nova.elotl.co/cluster.zone=us-west
-2d
```

Spread/duplicate vllm1 ns for LLM deploy; Set spread/duplicate/labeled policy for ns workloads

```
$ export RAYCLUSTER_NAMESPACE=vllm1
$ envsubst < ${SKYRAY_PATH}/deploy-scripts/namespace.yaml | kubectl apply -f -
$ envsubst < ${SKYRAY_PATH}/policies/rayservicespreadypolicy.yaml | kubectl apply -f -
```

Deploy RayService to serve LLM model mosaicml/mpt-7b-chat using vLLM to vllm1 namespace

```
$ kubectl apply -f ${SKYRAY_PATH}/luna-llm-serve/ray-service.llm.yaml -n ${RAYCLUSTER_NAMESPACE}
$ kubectl get rayservice.ray.io/llm-model-serve -n ${RAYCLUSTER_NAMESPACE} -o
jsonpath='{.metadata.annotations.nova\.elotl\.co/spread-onto}' | awk {'print $NF'}
workload-cluster-1::Duplicate
$ export TARG_CLUSTER=workload-cluster-1
```

LLM Model Serving w/No downtime K8s upgrade, Con’t

Validate serving LLM model from vllm1 namespace via query.py

```
$ kubectl --context=${TARG_CLUSTER} port-forward service/llm-model-serve-serve-svc 8000:8000 -n ${RAYCLUSTER_NAMESPACE}
$ python query.py
Type your query here: what year did the titanic sink? The Titanic sank in April 1912.
```

Get copy of Nova cluster CR

```
$ kubectl get cluster workload-cluster-1 -o yaml > workload-cluster-1.yaml
$ cp workload-cluster-1.yaml workload-cluster-1-upd.yaml
```

Edit cluster CR copy to change the Nova cluster name, the K8s cluster name, and the K8s version

```
$ diff workload-cluster-1.yaml workload-cluster-1-upd.yaml
11c11
<     kubernetes.io/metadata.name: workload-cluster-1
—
>     kubernetes.io/metadata.name: workload-cluster-1-upd
17c17
<   name: workload-cluster-1
—
>   name: workload-cluster-1-upd
21,23c21,23
<   kubernetescluster: cpu-us-west-2
<   kubernetesversion: "1.30"
<   name: workload-cluster-1
—
>   kubernetescluster: cpu-us-west-2-upd
>   kubernetesversion: "1.31"
>   name: workload-cluster-1-upd
```

“No Downtime K8s upgrade” via clone-upgrade/spread/validate

Apply resulting updated CR copy

```
$ kubectl apply -f workload-cluster-1-upd.yaml
```

New cluster CR label matches LLM spread/duplicate policy; Nova creates cluster for placement

```
$ kubectl get rayservice.ray.io/llm-model-serve -n ${RAYCLUSTER_NAMESPACE} -o
jsonpath='{.metadata.annotations.nova\.elotl\.co/spread-onto}' | awk {'print $NF'}
workload-cluster-1,workload-cluster-1-upd::Duplicate
$ kubectl get clusters
NAME                    K8S-VERSION   K8S-CLUSTER        REGION      ZONE        READY   IDLE    STANDBY
workload-cluster-1      1.30          cpu-us-west-2      us-west-2   us-west-2d  True    False   False
workload-cluster-1-upd  1.31          cpu-us-west-2-upd  us-west-2   us-west-2d  True    False   False
```

User validates serving LLM model from vllm1 namespace on the upgraded cluster via query.py

```
$ export TARG_CLUSTER=workload-cluster-1-upd
$ kubectl --context=${TARG_CLUSTER} port-forward service/llm-model-serve-serve-svc 8000:8000 -n ${RAYCLUSTER_NAMESPACE}
$ python query.py
Type your query here: how many days are in March? There are 31 days in March.
```

User removes label from old cluster -> idle; JIT deletes after a while, user can then delete its CR

```
$ kubectl label cluster workload-cluster-1 cluster.type-
<SNIP>
$ kubectl get clusters
NAME                    K8S-VERSION   K8S-CLUSTER        REGION      ZONE        READY   IDLE    STANDBY
workload-cluster-1-upd  1.31          cpu-us-west-2-upd  us-west-2   us-west-2d  True    False   False
```

Scenario: RAG+LLM Serving w/RayService for LLM

For RAG+LLM serving w/RayService for LLM, desired behavior is "**select cluster for task**"

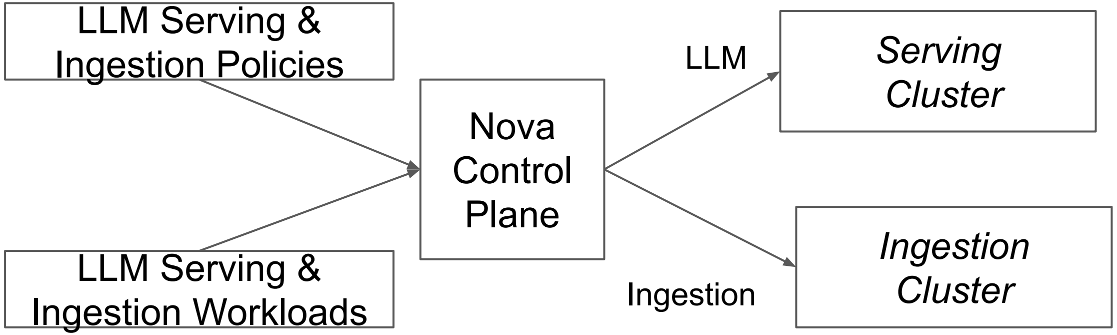
- **Nova cluster selection policy matches task namespace w/cluster for task**
- **CompoundAI example that uses KubeRay and non-KubeRay components**
- **Retrieval Augmented Generation ingests data into vectordb for context lookup during serving**

Ingestion job placed on JIT ingestion cluster comprised of static CPU-only cloud resources

- LangChain doc loaders/splitters, HuggingFace embeddings, FAISS vectorDB; pickled for S3 upload

LLM serving and RAG+LLM serving placed on serving cluster, which is dynamic

- LLM serving: RayService deployed by KubeRay w/autoscaler
- RAG+LLM serving: LangChain+FASTAPI+FAISS vectorDB S3 download/unpickle



On-demand cloud resources
w/a few static nodes running KubeRay
w/autoscaler & Luna cluster autoscaler

Static CPU-only cloud resources
JIT cluster using delete/create standby

RayService Serving LLM model with RAG, Start

Workload clusters set up

```
$ kubectl get clusters
NAME                 K8S-VERSION   K8S-CLUSTER   REGION      ZONE         READY   IDLE   STANDBY
ingestion-cluster    1.30          cpu-cluster   us-west-2   us-west-2b   True    True   False
workload-cluster-1   1.30          gpu-cluster   us-west-2   us-west-2d   True    True   False
```

EKS workload-cluster-1 static nodes

- `2x m5.large: 2 CPUs, 8 GB memory`

EKS ingestion-cluster static nodes

- `2x m5.large: 2 CPUs, 8 GB memory`

workload-cluster-1 labeled for rayservice serving and for model rag serving

```
 $ kubectl label cluster workload-cluster-1 cluster.type=rayservice
 $ kubectl label cluster workload-cluster-1 nova.elotl.co/cluster.modelragllmserve=true
```

RayService Serving LLM model with RAG, Continue

Place LLM; KubeRay creates head, Ray Autoscaler creates worker, Luna allocates nodes

```
$ export RAYCLUSTER_NAMESPACE=serving
$ envsubst < ${SKYRAY_PATH}/deploy-scripts/namespace.yaml | kubectl apply -f -
$ envsubst < ${SKYRAY_PATH}/policies/rayservicespreadpolicy.yaml | kubectl apply -f -
$ kubectl apply -f ${SKYRAY_PATH}/luna-llm-serve/ray-service.llm.mpt-7b-chat.autoscale.yaml -n ${RAYCLUSTER_NAMESPACE}
```

Validate LLM model running on workload-cluster-1

```
$ kubectl --context=workload-cluster-1 port-forward service/llm-model-serve-serve-svc 8000:8000 -n serving
$ python query.py
Type your query here: what is the longest bridge in the world?
The longest bridge in the world is the Akashi-Kaikyo Bridge in Japan. It is a suspension bridge that spans a total length of 20.8
kilometers (12.6 miles) and connects the city of Kobe in Japan to the island of Honshu.
```

Run ingestion job on ingestion-cluster

```
$ export MODEL_INGESTION_CLUSTER=ingestion-cluster
$ export MODEL_NAMESPACE=serving
$ envsubst < ${GENAI_PATH}/demo/llm.vdb.service/specificclusterpolicy.yaml | kubectl apply -f -
# export AWS_ACCESS_KEY_ID=<censored>, AWS_SECRET_ACCESS_KEY=<censored>, VECTOR_DB_INPUT_TYPE=text-docs.
#  VECTOR_DB_INPUT_ARG=mini-rag-wikipedia-input, VECTOR_DB_S3_BUCKET=selvi-faiss-vectordbs.
VECTOR_DB_S3_FILE=anne-rag-wikipedia.pkl
$ envsubst < ${GENAI_PATH}/demo/llm.vdb.service/createvdb.yaml | kubectl apply -n ${MODEL_NAMESPACE} -f -
```

Place RAG+LLM serving service on workload-cluster-1

```
$ envsubst < ${GENAI_PATH}/demo/llm.rag.service/spreadacrossclusterset.yaml | kubectl apply -f -
$ export MODEL_LLM_SERVER_URL=http://llm-model-serve-serve-svc.serving.svc.cluster.local:8000
$ envsubst < ${GENAI_PATH}/demo/llm.rag.service/chat-serverag1lmpluslb.yaml | kubectl apply -n ${MODEL_NAMESPACE} -f -
```

"Select cluster for task" via Nova policies and JIT, Ray & Luna autoscaler

Run RAG+LLM query, RAG data was rag-mini-wikipedia

```
$ curl -X GET
"http://a8524e9bf88034df4b01095a0949752c-549500553.us-west-2.elb.amazonaws.com/answer/what%20are%20the%20two%20types%20of%20elephants%20in%20Africa?"

{"question":"what are the two types of elephants in Africa","answer":"The two types of elephants in Africa are the African
Forest Elephant and the African Savanna Elephant."}
```

And note that the idle ingestion cluster entered standby and was deleted from the cloud

```
$ kubectl get clusters
NAME                 K8S-VERSION   K8S-CLUSTER   REGION      ZONE         READY   IDLE    STANDBY
ingestion-cluster    1.30          cpu-cluster   us-west-2   us-west-2b   False   True    True
workload-cluster-1   1.30          gpu-cluster   us-west-2   us-west-2d   True    False   False
```

As usual, when done, cleanup is easy via namespace deletion

```
$ kubectl delete ns serving
```

Again, multi-cluster deployment as simple as single cluster deployment

Talk Outline

SkyRay Overview

SkyRay Operation

SkyRay Example ML/AI Use Cases

Conclusion

Conclusion

SkyRay seamlessly extends Kuberay for multi-cluster scheduling scenarios

- **Reduces launch-time** of critical ML/AI workloads by scheduling on available GPUs
- **Increases efficiency** by directing experimental ML/AI jobs to sunk-cost clusters
- **Manages costs** via policies that select GPUs with the desired price/performance
- **Enhances robustness** by supporting prioritized cluster use from multiple cloud vendors
- **Facilitates K8 cluster maintenance** with no workload downtime K8s upgrade
- **Handles scheduling compound AI jobs** with KubeRay models and with non-Ray components

We want SkyRay to handle your additional desired scheduling outcomes as well!

- And BTW Nova's flexible scheduling policies have been applied to various domains, including Managing LLM+RAG deployments, Performing cloud-agnostic gitops, Handling DR failover

If you'd like to try SkyRay, please download free trial versions of Nova and Luna. ELOTL

Thanks! Questions?

🙏