India
Timezone: Indian Standard Time (UTC+5:30)

Project Idea for Volcano

LFX Project proposal for Volcano

Personal Details and Contact Information

Full Name-	**Ronak Raj**
Github username -	https://github.com/RONAK-AI647
Email -	codeitronak226277@gmail.com

Synopsis

Understanding how the Volcano scheduler makes decisions, verifying plugin correctness, and measuring performance are essential yet challenging tasks—especially when introducing new features. Running a full Kubernetes cluster and simulating complex workloads for analysis is resource-intensive and slow. Unlike Kubernetes' kube-scheduler-simulator, Volcano lacks a lightweight, transparent, and dedicated tool to test and debug its scheduling logic. This proposal aims to bridge that gap by developing the Volcano Scheduler Simulator (VSS).

VSS will replicate Volcano's core scheduling logic—including queue selection, filtering, scoring, and binding—and provide detailed insights into scheduling decisions. The simulator will support Volcano-specific features like gang scheduling, queue policies, and topology-aware scheduling. It will enable developers to debug scheduling decisions, experiment with configurations, and evaluate performance efficiently.

What do we need?

After we successfully establish Kube-scheduler-simulator , the volcano scheduler will easily be able to:

- **Replicate Volcano's Scheduling Stages: Queue selection, filtering, scoring (with plugins), and binding.**
- **Support for Input Files: Accept YAML/JSON definitions for cluster state (nodes, pods, queues, jobs) and scheduler configuration (plugins, policies, resource models).**
- **Accept input files (YAML/JSON) for:**
- **1)Cluster state: nodes, pods, queues, job**
- **2)Scheduler configuration: plugins, policies, resource models.**

- **Provide detailed outputs:**

    - **Scheduling decisions (pod-to-node assignments).**
    - **Logs of filtering and scoring stages.**
    - **Explanations of why pods are unschedulable.**
    - **Resource utilization breakdowns**

Benefits for the Community

- **Enhanced Debugging: Simplifies debugging of complex scheduling decisions without requiring a full cluster.**
- **Faster Experimentation: Allows users to simulate and evaluate different scheduling policies quickly.**
- **Resource Efficiency:No need for big, costly test clusters.**
- **Transparency: See exactly *why* pods are scheduled the way they are.**
- **Support for Volcano Features: Simulates advanced scheduling features like gang scheduling, queue policies, and topology-aware scheduling.**
- **Community Enablement: Encourages innovation, experimentation, and knowledge-sharing within the Volcano ecosystem.**

Project Goals

Goal 1: Build the Simulator Core
Establish a kube-schedule-simulator that replicates **Volcano's scheduling logic** – including queue selection, filtering, scoring, and binding – with detailed insights on scheduling decisions.

Goal 2: Provide Easy-to-Use Interfaces
Design a **simple CLI/API** to accept inputs like **nodes, pods, and plugin settings**, and simulate **Volcano-specific features** such as gang scheduling and queue policies.

Goal 3:.Enable Debugging and Learning: Generate reports, logs, and explanations for scheduling behavior.

Goal 4:**Provide Web UI:** Enhance accessibility through a user-friendly interface.

Goal 5:**Support Advanced Features:** Gang scheduling, queue prioritization, topology-awareness, and performance metrics.

Goal 6:**Deliver Comprehensive Documentation:** Include tutorials, examples, and best practices.

Phased Approach (Implementation Plan)

Phase 1: Foundations (Setup and Planning)

- **Analyze Volcano's scheduler architecture(in/simulator ; in/web), stages, and plugin system.**
- **Design simulator architecture and core modules.**
- **Set up project repository, CI/CD pipelines, and testing baseline.**

Phase 2: Core Simulation Engine(adding plugins of scheduling Framework and add extenders plugin-extender)

- **Implement key scheduling stages:**
  - **Queue selection**
  - **Node filtering**
  - **Node scoring (using plugins)**
  - **Pod binding**
- **Support Kubernetes objects: Nodes, Pods, and Volcano-specific configuration. We just keep in mind that the simulator has its own configuration.**

Phase 3: Output, Logging, and Reporting

- **Develop detailed logging:**
  - **Nodes considered and filtered**
  - **Scoring details**
  - **Final pod-node assignments**
- **Support CLI/JSON outputs**
- **Add a Web UI layer for visualizations.**

  Web Ui is the easiest way to check the schedulers behaviours. It provides a nice table view for the scheduling result . It also comes with a yaml editor to create/edit resources. We a-can see the detailed result like each filter plugin , each score plugin , and the final score .Taking snapshots of resource states for sharing or later use is made easier.

Phase 4: Advanced Features

- **Add performance metrics:**
  - **Scheduling latency**
  - **Throughout analysis**
- **Support more complex scenarios:**
  - **Gang scheduling**
  - **Queue prioritization**
  - **Topology-aware scheduling**

  Optional integration with Kwok for simulating Kubernetes cluster states.

Phase 5: Documentation and Community Engagement

- **Write detailed documentation, tutorials, and example use cases.**
- **Engage the community for feedback and iterative improvements.**
- **Create demo scenarios for users to test different scheduling strategies.**

Phase 6: Release and Maintenance

- **Finalize testing and validation.**
- **Publish an open-source release.**
- **Provide ongoing maintenance, support, and feature updates based on community feedback.**

Kubernetes-scheduler-simulator

How does the simulator work?
We need to know that the simulator works with HTTP server , while in our project we also wish to launch debuggable-scheduler and controller -manager and kube -apiserver outside.and we will be using KWOK for this purpose. At first

1. **The users request creating resources by communicating with the kube-apiserver of KWOK via any clients like (kubectl, k8s client library or Web UI).**
2. **The scheduler schedules a new pod**
3. **The results of score/filter plugins are recorded**
4. **The scheduler binds the pod to a node**

We can integrate our scheduler into the simulator by :
1) To add our custom scheduler plugins
2)Also, if you just want to use your KubeSchedulerConfig while using default plugins, you don't need to follow this page. Check out simulator-server-config.md instead.

Challenges

1) Complex Logic
Volcano's scheduler has many steps—queueing, filtering, scoring, binding—and plugins can modify each other's behavior. Simulating all this accurately yet simply is tricky.

2)Plugin Interactions
Plugins like gang scheduling and node scoring must work together in the right order. Recreating these interactions without overcomplicating the simulator is a challenge.

3) Lightweight Focus
The simulator must stay lightweight—not a full Kubernetes cluster—but still useful enough to debug and test key scheduling scenarios.

4) Accuracy and Testing
You'll need to validate the simulator's output against real scheduling results, which requires good test cases and comparisons.

5) Community Scope
Once released, the community may ask for more features. It's important to define scope clearly and avoid scope creep.

We will be having answers to these questions by end:

1)How to speed up simulation? For example, how can a 20-hour pod run be simulated in seconds?
2)What features of Kwok to leverage? Can Kwok fully replace the need for time simulation and API server simulation?.

Solution :

Integration with Kwok

Kwok (Kubernetes WithOut Kubelet) is a lightweight Kubernetes cluster simulator. We propose integrating Kwok into the simulator to:

- Simulate **large cluster environments** (e.g., 1000+ nodes) without the overhead of a real apiserver/kubelet.
- Accelerate time to **simulate long-running workloads** without delays.**Where feasible, we will:**
- Reuse Kwok's capabilities for API server and node simulation.
- Avoid duplicating simulation efforts for node/pod state.
- Focus on enhancing the scheduler logic simulation specific to Volcano
- **The Volcano Scheduler Simulator** is the **tool we are proposing to build**.
- **Kwok** is an **optional dependency or utility** that can be used **alongside** the simulator, but it is **not the simulator itself. Reference** kwok

Expected Output

- **A lightweight CLI/API tool to simulate Volcano's scheduling logic.**
- **Human-readable logs and detailed JSON/CSV reports.**
- **Clear explanations of pod scheduling decisions, filtered nodes, and scores.**
- **Support for Volcano features: gang scheduling, queue policies, topology-awareness.**
- **Optional performance metrics: latency, throughput.**
- **Tutorials, examples, and community-ready documentation.**

Why me?

I am deeply passionate about Kubernetes, cloud-native systems, and open-source development. My technical background in Golang, distributed systems, and Kubernetes architecture aligns perfectly with this project. I have already explored the Volcano codebase, participated in relevant discussions, and demonstrated a strong understanding of scheduler internals, plugin mechanisms, and simulation frameworks.

I am a fast learner and a dedicated contributor, eager to collaborate with the community, learn from mentors, and deliver impactful solutions. I am driven by the desire to make complex systems like Volcano more transparent and accessible for developers, researchers, and users. Given the opportunity, I will ensure this project meets high-quality standards, with clear documentation, tests, and a user-focused approach.

I truly believe that dedicated new contributors deserve opportunities in open source, especially when they're ready to commit fully and treat the project as a primary focus rather than a side task — something that aligns with the values LFX promotes. I currently have several open issues and pull requests in the Volcano Dashboard repository, and I'm fully committed to completing them and contributing my best to the project.

ISSUES:

- **https://github.com/volcano-sh/dashboard/issues/146**
- **https://github.com/volcano-sh/dashboard/issues/127**
- **https://github.com/volcano-sh/dashboard/issues/124**
- **https://github.com/volcano-sh/dashboard/issues/120**
- **https://github.com/volcano-sh/dashboard/issues/119**
- **https://github.com/volcano-sh/dashboard/issues/90**
- **https://github.com/volcano-sh/dashboard/issues/8**

PULL REQUESTS:

- **https://github.com/volcano-sh/dashboard/pull/137(prepare DCO sign off)**
- **https://github.com/volcano-sh/dashboard/pull/136(create docker-compose.yml)**
- **https://github.com/volcano-sh/dashboard/pull/126(husky set up)**
- **https://github.com/volcano-sh/dashboard/pull/125(pull request template)**