Easier JupyterLab Instances for HPC Users

Currently, HPC users wanting to use a JupyterLab instance have one of two options, and both involve tunneling the JupyterLab session over an SSH connection. The easiest method is to launch the JupyterLab instance on a front-end node, but the downside to this is that the front-end nodes are shared amongst users, so self-restraint of consuming computational resources in the JupyterLab instance is required on the user’s part. The other is to tunnel the JupyterLab instance from an interactive SLURM allocation. This second option is cumbersome as it requires a "double tunnel" over two SSH connections, and the issue of lack of immediacy if the interactive job can’t be granted at submission time. We would like to make it much easier for HPC users to have JupyterLab instances on our HPC resources. First, we opted to use Apache Mesos, a cluster manager, to host JupyterLab instances and run their corresponding jobs on its worker nodes. However, due to lack of support for Apache Mesos, we’ve recently opted to use a JupyterLab platform powered by Jupyter Enterprise Gateway (JEG) on a Kubernetes cluster. JEG provides optimal resource allocations by enabling Jupyter kernels to be launched in their own Kubernetes pods, allowing notebooks to use minimal resources. By default, Jupyter runs kernels locally - potentially exhausting the server of resources. By leveraging the functionality of Kubernetes, JEG distributes kernels across the compute cluster, dramatically increasing the number of simultaneously active kernels. In this talk, we make the case for a JEG-on-Kubernetes system to provide JupyterLab sessions for our HPC users.

LA-UR: LA-UR-20-25758