Tuning Performance and Scalability of KNIME Workflows

Iris Adä
Iris.Adae@knime.com

Phil Winters
Phil.Winters@knime.com

Summary

With KNIME Analytics Platform you are spoiled for choice in creating great data science. You can of course build everything using KNIME nodes, but other choices exist. Want a workflow that uses available in-DB capabilities then moves to a production Apache Spark setup - while at the same time using a special Google service before comparing a KNIME Random Forest to an H2O Random Forest, automatically choosing the correct model, which then gets applied to your favourite CRM so that the new score is placed back into the CRM? No problem in KNIME.

This article provides background around choice and tuning options, as well as an approach and sample workflows for determining the “right” combination for your specific requirements.

Table of Contents

Introduction

With KNIME Analytics Platform you are spoiled for choice in creating great data science. You can of course build everything using KNIME nodes, but other choices exist. Want a workflow that uses available in-DB capabilities then moves to a production Apache Spark setup - while at the same time using a special Google service before comparing a KNIME Random Forest to an H2O Random Forest, automatically choosing the correct model, which then gets applied to your favourite CRM so that the new score is placed back into the CRM? No problem in KNIME.

Or do you want to use AWS as well as Azure ML services along with KNIME nodes to provide a focused Guided Analytics application to end users? Again, a straightforward build of nodes in a workflow which, when finished, can be one click deployed to KNIME Server and made instantly available via the KNIME WebPortal. The same minimal effort is needed to create a RESTful Webservice on KNIME Server, which enables you to make all your new achievements callable from within other applications.

The choices are extremely wide. While KNIME makes it straightforward to mix and match in an easy way, not all roads that lead to Rome will be equal in terms of performance and scalability. One recent user request asked:

“I have built two workflows each containing six nodes, but in two different ways. When I run them with 100k, the performance is the same. When I run them with 10 million records, one takes five times as long as the other. Why? It’s still only six nodes”.

This article provides background around choice and tuning options, as well as an approach and sample workflows for determining the “right” combination for your specific requirements.

Choice and Tuning Options

There are four major areas that can be managed in order to obtain optimal workflow performance and provide scalability options.

1. KNIME setup and workflow options
2. Hardware and resource options
3. KNIME extensions (including other 3rd party packages)
4. Additional capabilities provided by KNIME Server such as remote executors

In this article, we will focus on the first three, which affect KNIME Analytics platform (and naturally its compliment - KNIME Server). A follow-up article will cover the additional KNIME Server performance and scalability options for those that have KNIME Server as well.

KNIME Setup and Workflow Options

There are options to choose from in the way KNIME itself is setup as well as different workflow configuration options.

- KNIME Analytics Platform setup options include providing access to more memory, redirecting workspace usage to use available high-performance options such as SSD, or setting the size of tables held in memory
- Workflow setup options include capabilities such as streaming sequences of nodes (so that no intermediate tables are written) as well as workflow control nodes for packaging workflow segments to take specific advantage of parallel capabilities

These very important topics are discussed in the blog article Optimizing KNIME Workflows for Performance.

Hardware and Resource Options

KNIME Analytics Platform as well as KNIME Server are exactly the same on all supported operating systems and hosting options, which means there is a lot of choice. Hardware and resource options include your choice of operating system – Windows, MAC or Linux. You can choose your preferred hosting option, which can range from standalone PCs and servers to cloud hosting options such as Azure and AWS. You also have the ability to combine all of these options. Along with all of these hosting options, there are also hardware factors to take into consideration such as memory, disk type, cores, as well as the availability of specialized hardware such as GPUs. KNIME can be set up to use all the available resources.

KNIME Extensions & 3rd Party Packages

KNIME supports an ever-growing range of 3rd party integrations. Highlights include support of over 70 native DBs, Python, R and Java, Apache Spark and H2O, Keras, Tenserflow and Kubernetes, as well as Azure, AWS and Google services. A complete list can be found on the KNIME Hub. Where possible, nodes are provided so that the at-times very different approaches of each of those extensions can be easily utilized within KNIME.

Virtually all these 3rd party extensions provide their own unique execution environment as well as data structures for data. KNIME takes care of the access or the setup to provide access to these environments. Note, however, that crossing from one extension to another can mean not only needing to tune the other execution environment but also moving data from one extension infrastructure to another. KNIME makes this extremely easy from a workflow building perspective by either automatically performing the conversion (as for Python, R and Java) or providing nodes to perform the conversion (for example between DBs and Apache Hadoop). For workflows that process a lot of data this means that performance effects can be incurred due to these “simple looking” nodes.

Performance and Scalability Approach

Given such huge choice in KNIME, there is no single “best” recommendation. Instead, the power of KNIME can be used to build various scenarios. These can then be executed and compared to choose the best for a given situation.

To support you in doing this, we recommend a six-step approach to evaluate and identify an optimal workflow configuration.

The Six Step Approach

Step 1: Define your workflow with KNIME nodes

A good first step is to create your workflow using KNIME nodes. You can work through each step of your workflow to get to the ideal workflow that will create and deliver the new insight in the appropriate form for your task. If you are accessing very large data sources, it may be more efficient to work on just a subset of the data during the development and prototyping phase.

At the end of this process, you could very well produce a workflow that already performs extremely well in KNIME by utilizing available hardware and infrastructure options. However, you may also decide to explore other options to see if further performance enhancements can be achieved or find options that could take advantage of existing corporate infrastructure, or follow specified corporate standards.

It is then helpful to go back and structure your workflow into logical groups that will later help in comparing and contrasting a workflow that has been implemented multiple times with different performance options. A

suggested series of steps might include: **Initialize, Load, Join, Transform, Feature Prep, Learn, Evaluate, Consolidate, Deploy** and/or **Interact.**

Step 2: Identify relevant capabilities

There may be good reasons to use additional KNIME extension capabilities. You may have other hosting options, such as a larger server or a cloud environment with additional resources. You may also have 3rd party extensions that already exist, such as databases, an Apache Hadoop infrastructure, or an additional package such as H2O, available within KNIME and which might increase performance. You might also be required to follow specific corporate standards in certain parts of the workflow.

In this step, we recommend identifying these capabilities and creating a table where the options are the rows in the tables and workflows steps are the columns across the top, as shown in Figure 1.

This graphic shows several scenarios that might be good for testing given the requirements of a specific workflow. Your specific workflow will have different scenarios defined.

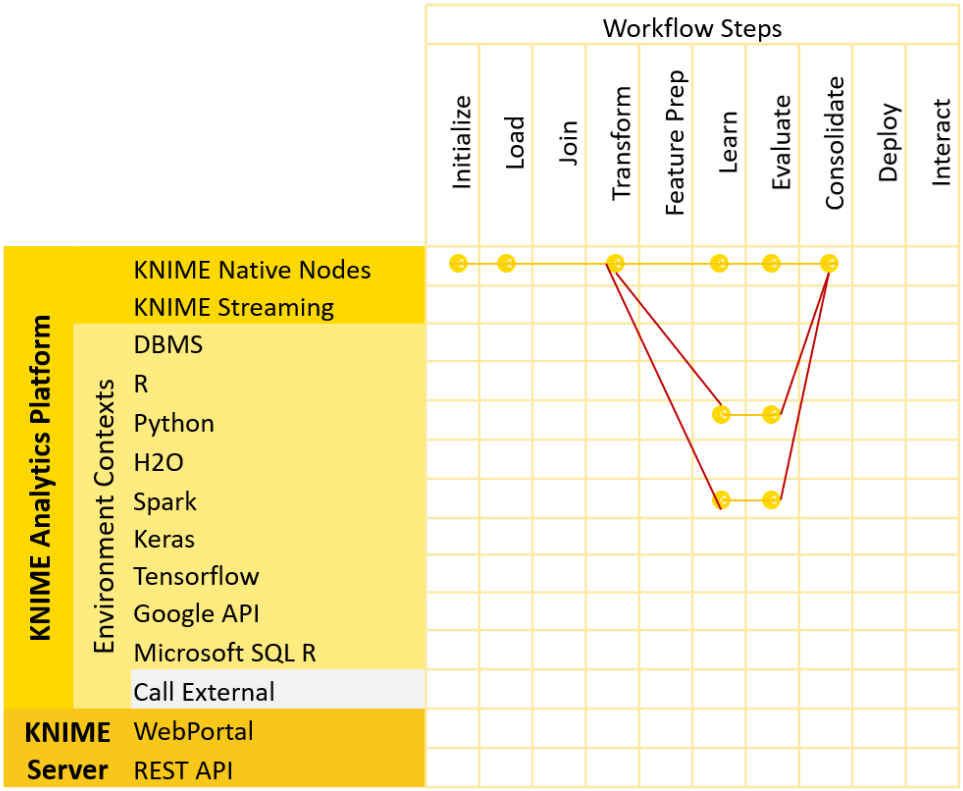

Figure 1.

Example workflow steps and available environment and extensions

Step 3: Define possible scenarios.

Not every capability will be suitable for every step in the process. This is where you would identify the sensible and available alternatives for each step. As a general rule of thumb, if large volumes of data are already in an infrastructure then you should try to do as many steps within that infrastructure as possible. For example, if all the data you need is already in a DB, then it may make sense to have a scenario that has your KNIME workflow using DB nodes to do the load, join, and transformation steps natively within the DB. Generally, the remaining steps may not be possible with most of the native DBs: at that point, you would consider continuing in KNIME; another example might be when all your data are already in Apache Hadoop and you have a large Apache Spark infrastructure available. A good scenario could be to perform as many of the steps using the KNIME Big Data extension nodes as possible within that environment.

Step 4: Deal with environment contexts

KNIME makes it as easy as possible to mix and match across extensions. But those extensions always bring with them their own environments. Python and R are automatically set up for you the first time you use them within a workflow. Many others, such as Apache Spark, Tenserflow, Keras, and H2O, need to be either started or accessed before they can be used. And of course, if you use REST API or other specialized calling nodes such as Google, they need to connect to an existing environment.

Just as KNIME itself has tuning options, each of these environments may also have tuning options that need to be set to ensure the environment is performing at its max. In some cases, KNIME gives you the ability to set some of those tuning parameters, for example you can use the KNIME H2O nodes to tell DL4J to use GPUs if available. But most tuning options for a particular extension will have been set (or need to be set) by the experts that maintain that specific environment.

For many organizations that are using these other environments heavily, this will have been done already. A good example here are the DB tuning options. Since KNIME calls these environments, it means that KNIME workflows also have the advantages of those tuning exercises.

If you are new to one of the extensions, KNIME makes it easy to try out and learn that extension. Good examples here include Apache Spark and H2O, where KNIME can spin up the infrastructure on your local machine for you to learn more about it. But in no way are these environments permanent or tuned and you would never want to include them in your performance and scalability testing.

Another consideration when moving between extensions is data. Most tools and extensions have their own data structures. To do anything in those tools, you need to have the data in those structures. Moving data between extensions or environments can be tedious. KNIME helps here by either doing it for you automatically (as in the case of R, Python and Java) or providing nodes that automatically convert between environments. But keep in mind, as mentioned above, every environment change will need time and might slow your workflow.

You will know this is happening as you modify your KNIME workflows away from native KNIME nodes and use the extension-specific nodes: in many cases, you will need extra nodes to convert the data. This is a wonderful way of joining desperate extensions; behind the scenes this means that extra resources are being required to do a data conversion that would not be required if an entire workflow was contained within one platform, e.g. all KNIME nodes.

One best practice is to move only those data that are required for further processing. For example, you may have all your data in a DB and can do all the data blending within that system. At the end, only a subset of columns and rows are needed for further processing and it is this subset that is then converted to a different format.

Step 5: Set environment and run

Now that you have evaluated your options, decided on a few scenarios, and thought about environment contexts, it's time to build workflows. Your goal is to have one workflow for each scenario, containing the appropriate nodes to run in your chosen combination of extensions. One of them should always be a workflow with native KNIME nodes.

Performance Benchmarking Nodes

Within KNIME, there is a robust series of benchmark nodes provided to capture the performance statistics of every aspect of a workflow. The concept is straightforward and shown in Figure 2.

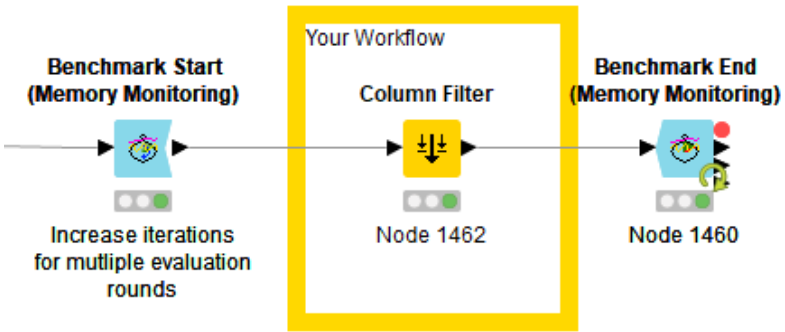

Figure 2.

Benchmarking nodes that wrap around the workflow to be measured

You begin your workflow with a **Benchmark Start (memory monitoring)** node. You end it with a **Benchmark End (memory monitoring)** node. When executed, the nodes capture statistics such as run time and memory usage. You can configure these nodes to penetrate down to the individual node level, even within a metanode or component. In this way, you are able to collect all relevant performance statistics.

These nodes are KNIME community nodes provided by the KNIME trusted partner, Vernalis Research Ltd. It is worth investigating the configuration options of the nodes. You can find these nodes on the KNIME Hub.

When these nodes are used, information about both individual node run times as well as total run time is collected and saved. They also capture the maximum memory used (including heap space) and the number of used cores.

Additional environment setting such as KNIME version and INI setting as well as operating system can be captured with the Extract System Properties node. You can also use your own labels and details to capture such information the running location (server or cloud, possibly including references to specific cloud instances) and well as information about whether SSDs and GPUs are used.

Step 6: Measure and compare

As soon as each scenario workflow has been built and you have inserted the benchmark nodes, you can run each workflow. In some cases, it is a good idea to run each workflow multiple times to get a better performance average for a given scenario.

When data volumes vary, possibly with - at times - a great increase data volumes, it pays to run each scenario on an ever-increasing volume of data. If you do not have enough data to do true volume testing, then KNIME's data generation nodes may prove helpful. Examples are available on the KNIME Hub under Data Generation and a full white paper of techniques can be found here.

Running your scenarios now produces a series of KNIME tables with performance statistics. You can then use KNIME to access and concatenate each of those tables to have one source to begin comparing and contrasting each scenario.

This is now a good point at which to include additional information about the scenarios. For example, you could add additional information to your workflow about your Cloud Instance or the cost of resources.

Below you can see what an example scenarios performance report can look like, in Figure 3:

Method	Hardware Environment	Number Rows	num Processors	XMXinG	heap Comitted MB	heap Usaged MB	Duration of Execution seconds	Accuracy	MS per record	Runtime Cost
Native KNIME	Laptop	10000	8	24	5020	582	6	0.605	0.64	0.000
Native KNIME	Laptop	100000	8	24	8429	1314	13	0.614	0.13	0.000
Native KNIME	Laptop	1000000	8	24	15702	6652	160	0.613	0.16	0.000
Native KNIME	AWS	10000	16	50	15657	945	6	0.606	0.62	0.124
Native KNIME	AWS	100000	16	50	18208	1373	13	0.609	0.13	0.262
Native KNIME	AWS	1000000	16	50	24759	6058	86	0.615	0.09	1.710
H2O in KNIME	Laptop	10000	8	24	5003	XXX	XXX	XXX	XXX	XXX
H2O in KNIME	Laptop	100000	8	24	8671	XXX	XXX	XXX	XXX	XXX
H2O in KNIME	Laptop	1000000	8	24	15911	XXX	XXX	XXX	XXX	XXX
H2O in KNIME	AWS	10000	16	60	15902	XXX	XXX	XXX	XXX	XXX
H2O in KNIME	AWS	100000	16	60	18246	XXX	XXX	XXX	XXX	XXX
H2O in KNIME	AWS	1000000	16	60	25000	XXX	XXX	XXX	XXX	XXX
Apache Spark	Laptop	10000	8	24	5077	XXX	XXX	XXX	XXX	XXX
Apache Spark	Laptop	100000	8	24	8805	XXX	XXX	XXX	XXX	XXX
Apache Spark	Laptop	1000000	8	24	16410	XXX	XXX	XXX	XXX	XXX
Apache Spark	AWS	10000	16	60	16113	XXX	XXX	XXX	XXX	XXX
Apache Spark	AWS	100000	16	60	18271	XXX	XXX	XXX	XXX	XXX
Apache Spark	AWS	1000000	16	60	24917	XXX	XXX	XXX	XXX	XXX

Figure 3.

Example of Scenario Performance report

A second scenario performance report could include factors such as the cost of having a more performant scenario. An example of that is shown here:

Method	Number Rows	Laptop	AWS	Speedup in MS	% Speadup	Cost of Speedup
Native KNIME	10000	6383	6185	197	3%	$ 0.12
Native KNIME	100000	13190	13116	74	1%	$ 0.26
Native KNIME	1000000	160103	85516	74587	47%	$ 1.71
H2O in KNIME	10000	xxx	xxx	xxx	xxx	xxx
H2O in KNIME	100000	xxx	xxx	xxx	xxx	xxx
H2O in KNIME	1000000	xxx	xxx	xxx	xxx	xxx
Apache Spark	10000	xxx	xxx	xxx	xxx	xxx
Apache Spark	100000	xxx	xxx	xxx	xxx	xxx
Apache Spark	1000000	xxx	xxx	xxx	xxx	xxx

Figure 4.

Example of Scenario Speedup Comparison report

Sample Workflows

To help you get up to speed, you can find a complete set of working examples on the KNIME Hub. Three scenarios are presented here: one using all KNIME nodes, one using Apache Spark and one using H2O nodes wherever possible.

An overall master workflow runs all three of these scenarios multiple times, each with an ever-increasing number of records.

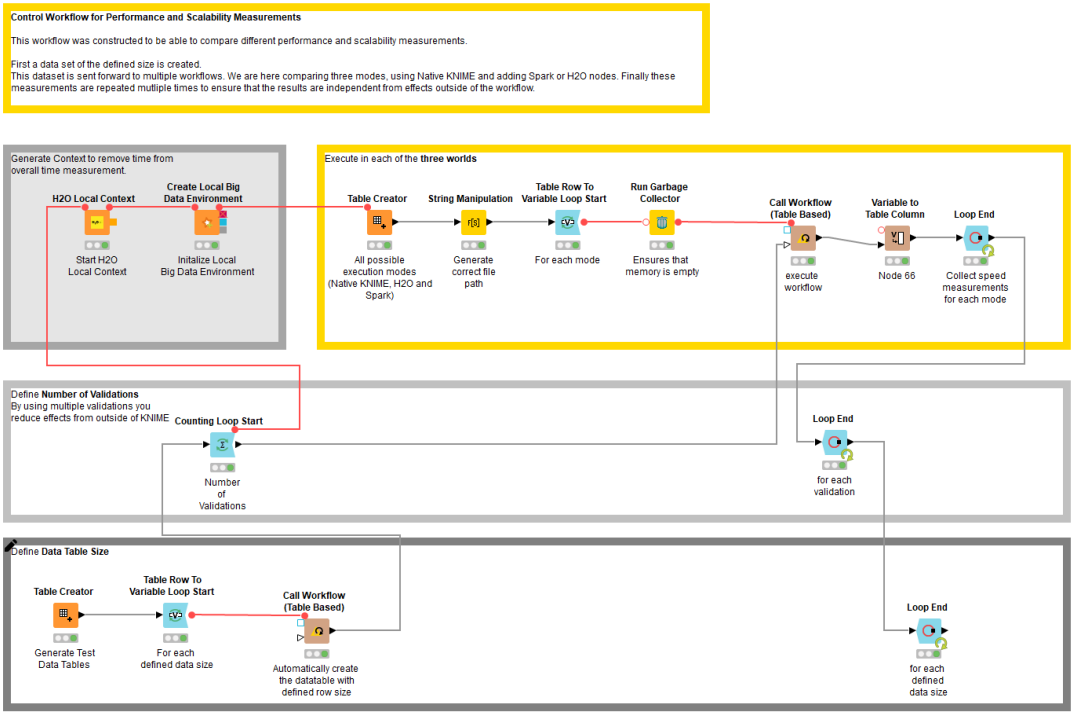

Figure 5.

Overall control workflow example for running three different scenarios

Conclusion

With this six-step process for testing the performance and scalability of scenarios in KNIME you have an extremely powerful way of making your choices and coming up with the best way to achieve a performant and scalable workflow for your specific data science challenge.