SAS1958-2015

Real-Time Risk Aggregation with SAS® High-Performance Risk and SAS® Event Stream Processing

Arvind Kulkarni, Albert Hopping, and Ling Xiang, SAS Institute Inc.

ABSTRACT

Risk managers and traders know that some knowledge loses its value quickly. Unfortunately, due to the computationally intensive nature of risk, most risk managers use stale data. Knowing your positions and risk intraday can provide immense value. Imagine knowing the portfolio risk impact of a trade before you execute. This paper shows you a path to doing real-time risk analysis leveraging capabilities from SAS® Event Stream Processing and SAS® High-Performance Risk. Event Stream Processing offers the ability to process large amounts of data with high throughput and low latency, including streaming real-time trade data from front-office systems into a centralized risk engine. High-Performance Risk enables robust, complex portfolio valuations and risk calculations quickly and accurately. In this paper, we present techniques and demonstrate concepts that enable you to more efficiently use these capabilities together. We also show techniques for analyzing High-Performance Risk data with SAS® Visual Analytics.

INTRODUCTION

In today's financial and commodities trading world, risk managers have the daunting task of controlling the risk of their trading books. They have to do this despite the fact that the makeup of their books is in constant flux due to continuous global trading. Traders are told to stay within their risk limits to keep their portfolios out of danger during adverse market conditions. Many times they do this without complete knowledge of how their trades are affecting the risk of the portfolio as a whole. The complex nature of financial markets and interactions between different instruments makes it difficult to determine how a trade will affect the current risk of the portfolio.

Trading shops work continually to develop new ways to calculate risk metrics more quickly. The quicker the risk values are given to the traders the more useful they are for providing a gauge of where their trading book stands. Using SAS® Event Stream Processing and SAS® High-Performance Risk you can develop a system that calculates risk metrics in real time and provides a way for traders to keep track of their trading book's risk as its composition changes.

This paper introduces you to High-Performance Risk and Event Stream Processing, two very flexible solutions offered by SAS. These solutions are only introduced at a high level, however, you can learn more via the references at the end of this paper. The sections following are meant to help you consider the design for real-time streaming of your risk data. You will also be presented with several common use cases and a few technical pointers.

BACKGROUD ON THE SOLUTIONS

SAS® Event Stream Processing provides the ability to perform real-time analytics on streaming data and SAS® High-Performance Risk serves as the risk engine, leveraging an in-memory distributed architecture to calculate risk metrics across the grid. The following provides some background on the two products.

SAS® EVENT STREAM PROCESSING

SAS® Event Stream Processing is a form of complex event processing that processes streams of data to analyze events in motion. Event Stream Processing stores queries through which it passes the streaming data to retrieve a constantly updated analysis as new events occur. The engine is designed with threaded processing in order to provide sub-millisecond processing and high-volume throughput. Event Stream Processing takes advantage of adapters built on a publish-and-subscribe model to read from data sources and stream data through a model.

Event Stream Processing model is built with source windows and derived windows that flow data through a query to filter and analyze the data into events of interest. The data flow model is a directed graph that is commonly called a Continuous Query, due to fact that the model uses a set of queries whose results

are constantly updated as new events are processed by the system. These results are stored in the derived windows which applications can subscribe to and receive the current snapshot of the query as well as updates to the query as new data streams through the model. SAS® High-Performance Risk is the application that subscribes to a window of the Event Stream Processing model in order to retrieve the results of the continuous query.

The publish-and-subscribe capabilities are integral to the use of Event Stream Processing in sending data to High-Performance Risk. You set up High-Performance Risk to subscribe to the model and you set up your data source to publish to the Event Stream Processing model. In this way the model is just an extract, transform, load (ETL) process to get the data into High-Performance Risk. The data that you send through Event Stream Processing becomes the portfolio data that High-Performance Risk uses to perform its analytics.

SAS® HIGH-PERFORMANCE RISK

SAS® High-Performance Risk is a pricing simulation and aggregation engine that can roll up risk calculations into hierarchies on-demand. High-Performance Risk is a product designed to reduce risk calculation times from days or hours down to minutes. It includes features such as the following:

- pricing of large, complex portfolios of financial instruments and risk positions
- market state generation and exposure calculations
- dynamic stress testing and scenario analysis
- calculation of Value-at-Risk (VaR) and potential future exposure (PFE)

High-Performance Risk keeps data in structures called risk cubes that are distributed across a commodity hardware based grid via Hadoop. The cubes contain the required data at the most granular level that can then be aggregated up in hierarchies determined by the end user on the fly.

BUILDING HIGH-PERFORMANCE RISK CUBES ON EVENT STREAM PROCESSING DATA

SAS® High-Performance Risk includes the ability to interface with SAS® Event Stream Processing in order to read in portfolio data during cube creation. It uses a combination of the ESPMODE option and the ESP statement within a PROC HPRISK call. Event Stream Processing server will have a window that is configured to stream data in a format that is agreeable to High-Performance Risk. From the server's perspective High-Performance Risk is just another subscriber to the publish-and-subscribe API. From High-Performance Risk's perspective, Event Stream Processing is just another way of reading in a portfolio.

An example of this transaction is as follows. High-Performance Risk asks the Event Stream Processing server for a schema so that it can map incoming data to the correct data types. The data is streamed to the grid nodes and appropriate calculations are performed during the cube building process. High-Performance Risk captures the data published by the Event Stream Processing window until it sends a disconnect signal which ends the session. Once the session is over, High-Performance Risk finalizes the cube making it ready of use.

The following is example code for High-Performance Risk to stream data from Event Stream Processing and create a risk cube:

```
proc hprisk
  task = runproject
  expprojlib = riskenv   /*points to libref containing a risk environment*/
  espmode = on                                  /*tells HPRisk to use ESP*/
  cube = "/sasshared/prod/cubes/subcube1"     /*cube descriptor to create*/
  gridpath = "/local/data/risk/hpespnq" /*where to store data on the grid*/
  ;
  esp host="orion123" port=22221
  posqryname="posqry" poswindow="poswindow"
  ; /*the ESP statement tells HPRisk how and where to listen to ESP*/
  performance nodes=10 nthreads=8; /*the number of nodes and threads used*/
```

```
    crossclassvars region portfolio strategy;
    /*cross-classification variables used for aggregation*/
  run; quit;
```

Building risk cubes can also be done in parallel. Multiple Event Stream Processing windows can publish data to their corresponding instances of High-Performance Risk. Figure 1 is an example of the architecture of the end-to-end flow of data.

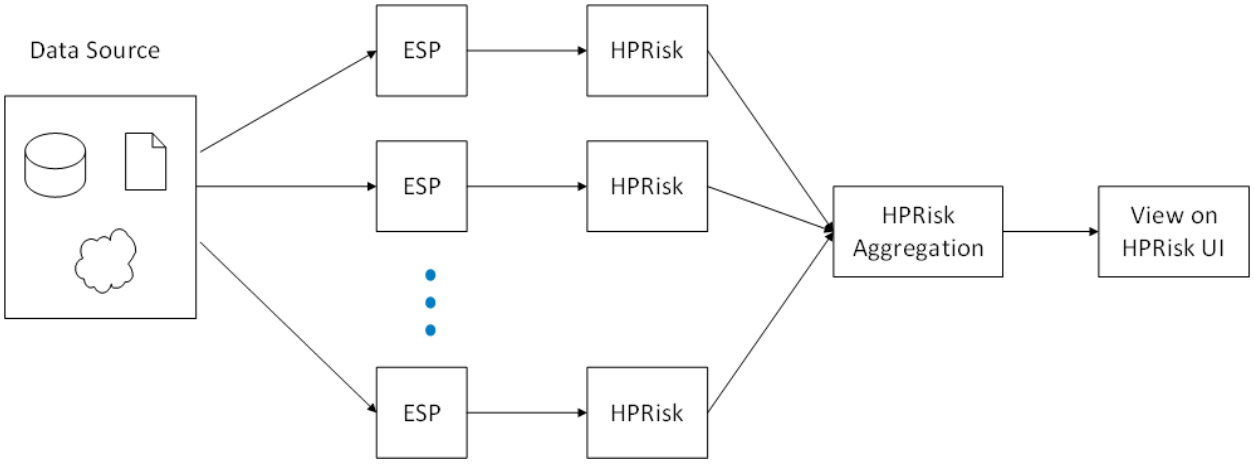


Figure 1. Architecture of System from Data Source(s) to Front End

Each call to PROC HPRISK as above will create one risk sub-cube, which contains portfolio, pricing, and other related High-Performance Risk data. This data is distributed across the grid over Hadoop. Each cube has a descriptor file created at the same time that contains information about the cube and points to its location on the grid. They can be analyzed and compared individually or aggregated by the thousands. High Performance Risk maintains its performance with the data divided into thousands of cubes the same as if all the data had been loaded into one large cube. This aggregation of the sub-cubes allows data to be trickled into the system throughout the day seamlessly to the end user.

REAL-TIME STREAMING DATA

The purpose of using SAS® High-Performance Risk in conjunction with SAS® Event Stream Processing is the ability for you to stream data in real time. Your risk calculations need updating as trading takes place throughout the day. When new data arrives it will stream through Event Stream Processing and into a High-Performance Risk cube. The sub-cubes can be structured in multiple ways and it depends on the design of the incoming data.

WHAT GOES INTO MY SUB-CUBE?

One of the main concerns about the data is whether trades may need to be updated throughout the day. This is important because sub-cubes cannot be altered once they have been created. Your system will need a way to ensure that the updated trade is included in an aggregate cube without double counting the original trade. This paper discusses two possible cube designs as below, both of which can handle updates if necessary.

Natural or Repeatable Groupings of Data

If there is a natural or repeatable grouping of data, you can use that grouping to determine what trades are part of the same cube. In order for the grouping to work, however, all trades within that group must be available to stream into the cube at the same time. For instance, if you decide to put an entire trade book into a cube then all trades in that book must be available for streaming when the cube is created. This design constraint is important if there are updates to trades at a later point as the entire cube needs to be replaced.

Pulsing Data into Separate Cubes

If there is no natural grouping that will work as the structure for the cubes, you can also pulse the data into separate cubes. Since cubes must be finalized before they can be used, it is necessary at some point to pause the streaming so that a cube can be built and restart it for the next cube. Event Stream Processing can pulse the cubes by time interval or by record count. The frequency of this interval is important. The further apart cube creation is, the longer users are delayed in seeing the data, however, making cubes too frequently is inefficient.

Cube Size

When picking the groups or pulse intervals the ultimate size of the sub-cubes should be considered. A particular concern is the inefficiency of making too many cubes too small. For example, cubes of only one position are possible, but not desirable. The time it takes to build a cube is roughly flat until a certain size and then it is roughly linear. This point of inflection varies with many factors such as the data structure, the network load, and the hardware. Creating cubes smaller than this point is not optimal, but may not be avoidable.

Determining what is too small depends on the specific data and hardware of a system. With this is mind, generally speaking, creating thousands of cubes to be aggregated may not cause any appreciable performance hit. However, dividing data into a hundred thousand cubes to be aggregated would be poor design and may reduce performance.

Intra-day Updates and Corrections

Your system might have to deal with data corrections over the course of the day. Depending on how your cubes are structured, there are multiple ways to handle corrections to trades. This paper goes over two possibilities based on the cube designs given in the previous section. The first is cube replacement where we remove the cube with the old trade and replace it with the cube that has the updated trade. Cube replacement gives rise to the need for a robust cube management system that controls which cubes are surfaced to the end user. The second is “delta trades” where the updated trade is added only as the delta from the original trade.

Updates by Cube Replacement

The cube replacement technique can be used when there is a grouping that dictates what trades are in each cube, but it is difficult when cubes are built from pulsed data. This is due to the fact that, when pulsing data, it is not easily determined what trades are actually included in each cube. A non-trivial detail of using the cube replacement technique is that when we remove a cube that includes a trade that has been updated, other trades that have not been updated are also removed. These trades need to be included in the new replacement cube in order to maintain consistency before and after the update. Therefore when an update occurs to one trade in a group the whole group needs to be streamed into a new cube, including trades that are unchanged. Using a repeatable grouping for cubes makes cube replacement easy.

Updates by “Delta Trade”

When the cubes are created by pulsing the data, the contents cannot be easily resent. Instead you can use a “delta trade”. This takes the updated trade and modifies it so that when it is added to the original trade it equals the new updated trade. This works because the user only views the data in aggregated views. If all of the cross classes are the same then the “delta trade” will always be aggregated with the original trade to show the updated amount. If more than the trade value changes, the original trade will need to be zeroed out and the new version of the trade added. With this technique no data is truly replaced, only added. “Delta trades” are best when very little data needs updating and it can be used for both group-based and pulsing-based cubes.

Star Schema

One of the best ways to load data faster is to load less data. By using a star schema for some of your variables, you can achieve several benefits. Conceptually, High-Performance Risk uses star schemas

similar to a database. The physical sub-cubes act as the fact table. The dimension or star data can be stored in Hadoop and is joined to the fact data when queried.

Choosing to omit some of your variables from the fact table and instead include them in star tables, via a schema statement, means there is less data Event Stream Processing needs to send to High-Performance Risk. The result can be faster data loads, less disk and memory utilization, and more efficient risk cubes. Using star schema also allows for the star tables to be updated on the fly with changes immediately surfaced in the High-Performance Risk Explorer.

While star dimension tables are supported as Hadoop, LASR, or dataset format, Hadoop is recommended. They can be loaded to Hadoop via a data step and a SAS libname. They need to share a single column as a key with the cube which must match in name and length. This libname needs to exist for the schema statement to run, but does not need to be auto-assigned anywhere. High-Performance Risk will use the HDFS path to find the star tables.

The Schema Statement

The schema statement is used in the PROC HPRISK call and can be used in either the cube build or a cube join. If many cubes are created, using the schema statement in the aggregate cube join is recommended. The libname of the star table does need to exist at the time the schema statement is used.

The following is an example of using a schema statement during cube aggregation:

```
proc hprisk
  cube="/sasshared/cubes/HPRisk_agg_cube"
  task=join
  jointype=aggregate
  subcubes=("/sasshared/cubes/sub1" "/sasshared/cubes/sub2"
"/sasshared/cubes/sub3")
  ;
  schema region data=hdfslib.regions crossclassvar regionID;
run;
quit;
```

This code creates the aggregate cube (HPRisk_agg_cube) of the listed sub-cubes while adding the schema variables. The primary key in this example is regionID. If the data in the star table hdfslib.regions changes, the changes will be seen in the explorer in real-time.

HOW DO I KEEP TRACK OF MY SUB-CUBES?

With what may be thousands of sub-cubes of new or corrected data a system is needed to track them. A simple inventory of the sub-cubes that exist is likely sufficient. The inventory of cubes can be used for sub-cube aggregation, deleting old cubes, and process monitoring.

Cube Management

Cube management tracks what cubes are current cubes that should be part of the aggregation that is surfaced to the user. When the system replaces a cube due to an updated trade it should not delete the original cube. The cube should still exist so that it is possible to roll back the risk to a time in the past. Thus the cube management process needs to track which cubes are currently part of the aggregation process and which cubes are old. This is not as much of a problem when using “delta trades”, but it is still important to keep track of the cubes that have been created as an input into the aggregation process.

The cube management process uses a separate dataset, the cube inventory, which has a record for every cube that exists. When new cubes are created a new record is appended to the dataset. Records are not deleted from the cube inventory. Instead, the aggregation process has logic to choose the correct cubes for the aggregate join.

Figure 2 is a representation of the cube inventory as new cubes are inserted and updated. It shows the records in the cube inventory and highlights the cube records that the aggregation process chooses for

the aggregate join. The letters represent the unique groups with which the sub-cubes are aggregated. You can see as a cube is inserted multiple times, the aggregation process only chooses the most up-to-date version of that sub-cube for the join. The orange shaded records represent the sub-cubes in the aggregate cube.

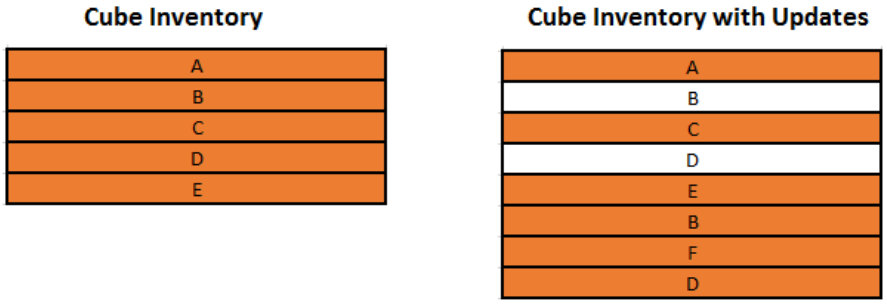


Figure 2. Representation of Cube Records that are part of the Aggregation

Data Reconciliation

It is useful to keep track of what data is sent to the system and what data is actually available in the system. Cubes are added to a cube inventory as they are built with High-Performance Risk. If your cubes are designed using a certain grouping then you know what data is currently available in the system. At the same time the system can keep track of what data is supposed to be in the system. One way to do this is to record the notifications that come whenever new data is available. There must be some logic inherent to the system so that it knows when to pick up new data either via a notification message or a scheduled process. This logic can be recorded to maintain a table of all the new data that should be inserted into the system. You can compare what data is available to what data should be available to determine if the current state is consistent with what is expected. You can use this data reconciliation report to help handle any discrepancies between what data you have received and what data is available on the front end to the user.

If your system is using pulsed data it is not clear what data each cube contains. This makes it harder to create a data reconciliation system. You can do a top level query on the cubes as they are created to collect information that can be helpful such as number of records. Additionally deeper queries on specific explanatory fields in the data will tell you more detailed information on what the cube contains.

SURFACING THE DATA

As data is streaming in you want to present this data to the user. The best way to do this is to surface a risk join cube, also called a virtual cube, in the High-Performance Risk Explorer. Virtual cubes are descriptor files that point to any number of sub-cubes. They are virtual because they only contain metadata that points to other cubes. There are two types of virtual cubes: aggregate cubes and comparison cubes. Comparison cubes are discussed later in this paper.

Virtual cubes do not contain risk data themselves. They are created quickly and use little disk space. Aggregate virtual cubes appear to an end user as a simple combination of their sub-cubes. Like any cube, a virtual cube is surfaced in the explorer by placing its descriptor file in the path for the web based explorer. Instead of making a copy of the descriptor, a symbolic link is used. This saves disk space and simplifies cube cleaning.

Here is an example of SAS using a `systask` command to create a link in the explorer for a cube using the Unix command `ln`:

```
systask command "ln –s /sasshared/cubes/agg1.rskcdesc
/sasshared/SASHighPerformanceRisk/3.3/cube_descs/price/curr_agg.rskcdesc";
```

As data is updated throughout the day, new aggregate virtual cubes are created. The pointer in the explorer can be repointed to the descriptor of this new aggregate cube. When the data in the explorer is refreshed, it will use the new aggregate cube with the newest data. This keeps users up to date.

USE CASES

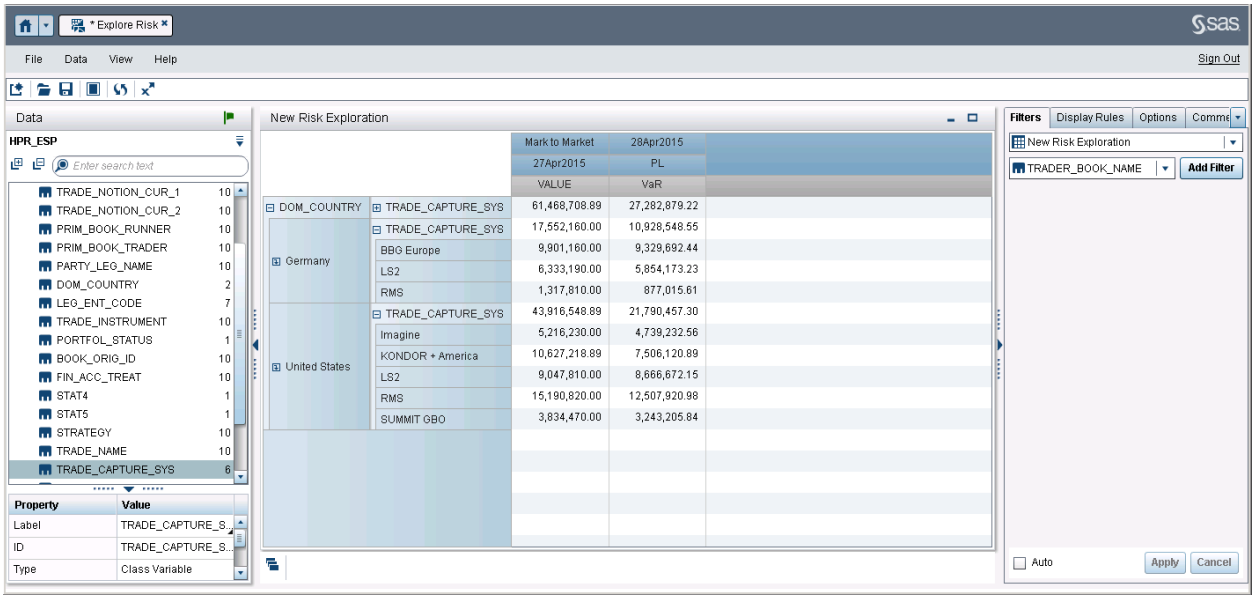
Some useful use cases utilizing SAS® Event Stream Processing and SAS® High-Performance Risk are presented below. Those cases are representative ones related to real-time risk aggregation. Certainly, other use cases can be developed according to specific needs.

VALUE-AT-RISK CALCULATION

VaR remains as one of most popular and widely used metrics to measure risk. With a flexible structure, High-Performance Risk supports two options to calculate VaR. First, a pricing library (a set of methods and functions) can be set up and registered in the risk environment. This pricing library can also be third-party library, e.g. FINCAD, FEA, and so on. High-Performance Risk generates or forecasts market states, in which financial instruments are priced by the pricing library. Then High-Performance Risk aggregates those prices into portfolio VaR. Second, instrument prices are calculated outside High-Performance Risk and those pre-priced values are imported into High-Performance Risk. The remaining steps in the calculation are the same as the first option.

For the first time, pairing High-Performance Risk with real-time streaming tools, like Event Stream Processing, enable you to calculate VaR on the streaming data in real-time fashion. This adds tremendous value for the users, especially a trader, to see how their risk is changing as new trades are introduced into their portfolio. Leveraging High-Performance Risk's super computing power, one can also quickly evaluate a deal from a risk perspective. Think of something like CVA (credit valuation adjustment), which notoriously demands complex computation to complete in a very short time period so as to be current with the ever-changing market condition. In addition, thanks to the lightning-fast performance, High-Performance Risk enables you to do an ad-hoc stress testing on its UI. This provide a quick way to evaluate your portfolio under stressful market condition.

Display 1 is a screen capture of the display window of the High-Performance Risk Explorer. It shows mark-to-market and VaR calculations for the portfolio as a whole and aggregated by sub-portfolio.



Display 1. Screenshot of High-Performance Risk Display of a Cube

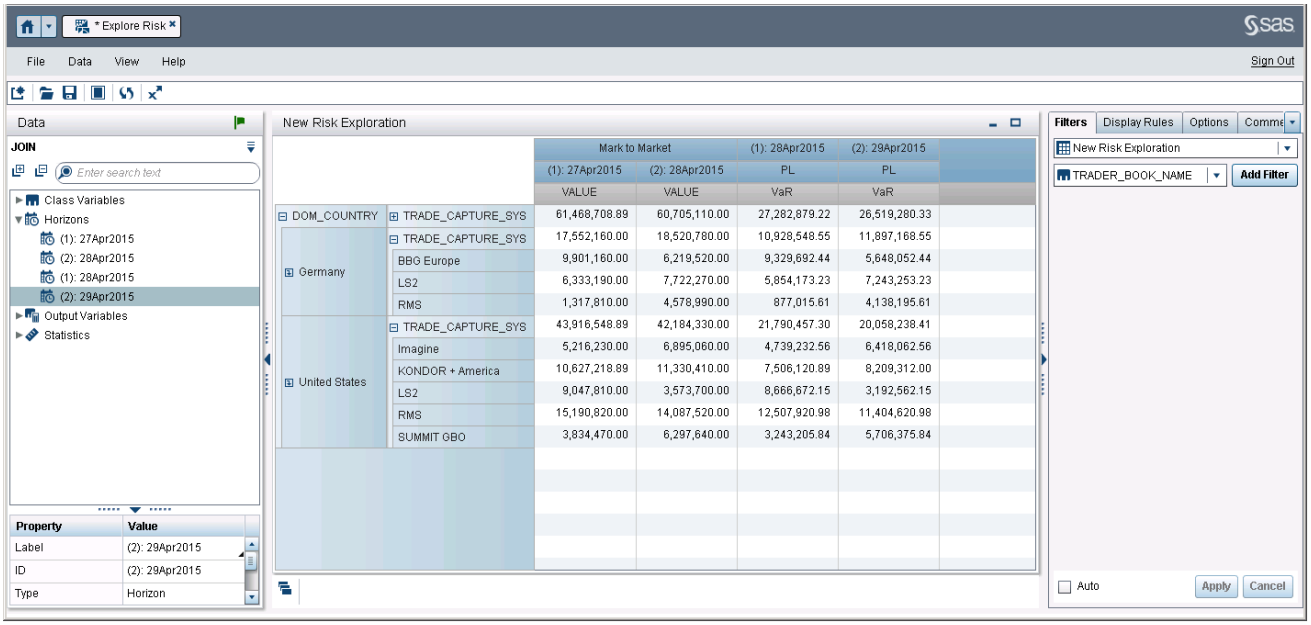
COMPARISON CUBE JOIN

In addition to aggregate joins, High-Performance Risk also performs comparison joins. While aggregate joins allow us to combine multiple cubes and treat them as a single cube, comparison cubes allow us to join multiple cubes and compare them. This is useful if you want to compare how the risk has changed. Taking it a step further, since the two cubes have the same cross-classes you can drill down and see how specific sub-portfolios have changed from one day to the next. The sub-cubes used in a comparison cube are queried, drilled, and sliced together, but not aggregated. Their results are shown side by side in the explorer for comparison.

This example code creates a virtual comparison cube that compares two aggregate cubes:

```
proc hprisk
  cube="/sasshared/cubes/HPRisk_comparison_cube"
  task=join
  jointype=comparison
  subcubes=("/sasshared/cubes/agg1" "/sasshared/cubes/agg2")
  ;
run;
quit;
```

Display 2 is a screen capture of a virtual comparison cube in the High-Performance Risk Explorer. You can compare the metrics between two days and see how the risk has changed.



Display 2. Screenshot of Comparison Cube

PAST VIEW CUBE

While the main purpose of this paper is a discussion of real time risk aggregation, it is also useful to see the risk calculations “as of” a certain time. The cube management process is important for this ability. The “as of” risk will include some cubes that are no longer in the aggregate cube and not include some new cubes that have arrived since that time. It is important, therefore, to keep cubes even if they are no longer relevant for the up-to-date aggregate cube. The cube aggregation process needs to be able to identify which cubes to join and which to ignore, and these change depending on the “as of” time of the join.

Figure 3 shows the representation of the cube inventory and the cubes selected past view aggregation. The cubes that arrive after the selected timestamp are ignored and the aggregation process selects the appropriate cubes as if grayed out records did not exist.

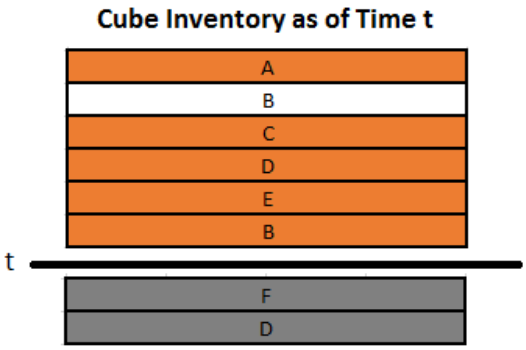


Figure 3. The Representation of the Cube Inventory Showing Aggregation as of Time t

MOVING RESULTS TO LASR

High-Performance Risk provides the capability to write out various types of result sets to the LASR Analytic Server (Orange 2013). Examples of data sets that can be written directly to LASR include:

- table with an observation for each instrument for each market state for which the portfolio is priced
- table containing output that was generated from credit scoring methods
- table of the mark-to-market value for the instruments in your portfolio

These data sets can be very large. However, they can be moved into LASR quickly because the action takes place almost solely within the grid nodes. Each node lifts its portion of the data from High-Performance Risk into LASR. This is called distributed mode.

The following is an example for writing the instrument MTM data to LASR. Note that the LIBNAME statement associated with `lasrlib` must include the SASIOLA argument in order to write to LASR.

```
proc hprisk
  cube="/sasshared/prod/cubes/HPRisk_cube"
  task=runproject
  outinstvals=lasrlib.instvals
  ;
run;
quit;
```

Result sets like VaR can also be writing to LASR. In this case of non-additive risk measures it is important to keep LASR and SAS® Visual Analytics users from summing them. This can be helped by sending results from HPRisk with the special pre-aggregated flag.

Once the data is in memory it can be quickly and easily analyzed and manipulated in the LASR Analytic Server using PROC IMSTAT. Reporting and exploration capabilities are also greatly expanded by Visual Analytics Designer and Visual Analytics Explorer respectively.

The data set is written directly to the SAS® LASR™ Analytic Server from the cluster node via memory, but only when the LASR Analytic Server nodes are the same as the SAS® High-Performance Risk cluster nodes. When High-Performance Risk cluster nodes are not the same as the LASR Analytic Server nodes, the data is written through the High-Performance Risk server head node instead. This situation could adversely affect performance. Further, the associated LIBNAME statement for the output data set must include the SASIOLA argument.

Temporary utilities files are created by each node when performing these writes to LASR. For the best performance, ensure these temporary files write locally to the node in a location with enough free space.

CONCLUSION

Pairing SAS® Event Stream Processing and SAS® High-Performance Risk can solve business challenges described above. Event Stream Processing streams real-time data into High-Performance Risk. High-Performance Risk processes (pricing, evaluation, aggregation, and so on) the data and produces a risk cube to be viewed and drilled through a user-defined hierarchy on the UI. Fast speed provides immense value to evaluate how a potential new trade would impact the risk profile of a portfolio in real-time. This enables a trading desk to quickly evaluate a trade and keep up with market conditions. All the risk cubes can be efficiently stored and managed for information retrieval later.

REFERENCES

Chen, W., Skoglund, J., and Iyer, S. (2014). "Effective Risk Aggregation and Reporting Using SAS." *SAS Global Forum 2014*, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings14/SAS193-2014.pdf.

Hatcher, D. (2015). "Real Time—Is That Your Final Decision." *SAS Global Forum 2015*, Cary, NC: SAS Institute Inc.

Orange, C., Christian S., and Erdman D. (2013). "Managing and Analyzing Financial Risk on Big Data with SAS High-Performance Risk and SAS Visual Analytics." *SAS Global Forum 2013*, Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings13/110-2013.pdf.

ACKNOWLEDGMENTS

We would like to thank Stacey Christian, Don Erdman, David Stonehouse, and Katherine Taylor for their help and understanding of SAS® High-Performance Risk in addition to their responsiveness to our suggestions for features. We would also like to thank Jerry Baulier and Vince Deters for their help with the inner workings of SAS® Event Stream Processing.

RECOMMENDED READING

- *SAS® Event Stream Processing 2.3: User's Guide*
- *SAS® High-Performance Risk Procedures Guide*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Arvind Kulkarni
SAS Institute Inc.
+1 (919) 531-1128
Arvind.Kulkarni@sas.com
http://www.sas.com/en_us/industry/banking/high-performance-risk.html
http://www.sas.com/en_us/software/data-management/event-stream-processing.html