# Understanding the Impact of the Software and Hardware Components that Drive Your Digital Press

Eric Worrall

Keywords: PDF, inkjet, RIP, performance, computer, benchmarking

**Abstract**

One of the most powerful features of digital print over traditional techniques is the ability to print every item differently. However, when you consider that digital printers are also becoming faster, wider and higher resolution the data rates required to drive the printer are also increasing in orders of magnitude. Workflows that use intermediate files on disk have already hit what we term a data rate barrier. One solution to this is to RIP and stream directly in real-time to the printhead electronics. However, as with most paradigm changes, it brings its own engineering challenges and drives new solutions.

A streaming RIP solution requires the right combination of software and PC performance and tight control over the PDF complexity and this needs to be done without adding to the overall bill of materials.

In this paper, we will discuss the complexities of selecting the software and hardware components at the heart of this real-time workflow. We will look at the trinity of factors that impact the data rate and how the complexity of the interaction has led to the creation of a dedicated benchmarking tool.

## Introduction

As digital printers advance to become faster, wider and higher resolution, supporting extended colorants and drop sizes, the raster data rates required to drive them at full-rated line speed is becoming a print industry barrier.

When you print many identical items, the data rate is relatively small, as the data is transferred once at the beginning and then many items are printed as fast as the

Global Graphics Software Ltd

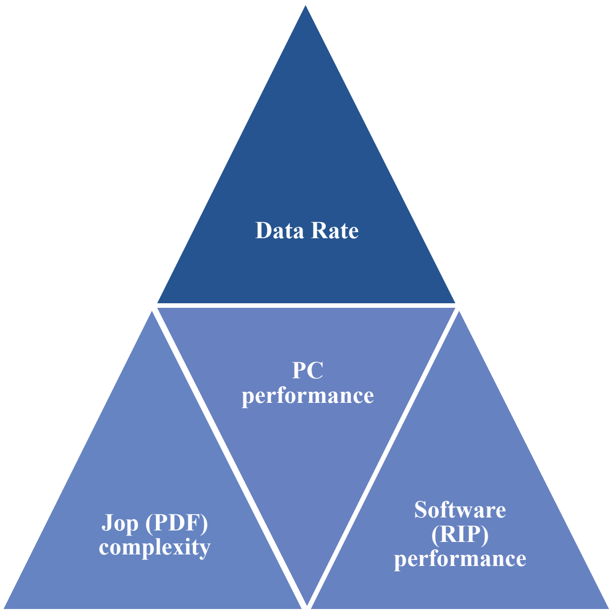
press can physically run. With an “every printed item is different” workflow, as is common with mass customization, the same amount of data is sent with each item and the rate of data required to keep the printer running is orders of magnitude higher than a traditional printer.

When required data rates were still small, you could RIP to disk ahead of time and then at some later stage submit the pre-RIPped rasters to the press with predictable performance. The main challenge with this is the disk reading and network speed. Choosing the PC and software to achieve average performance across the whole job is relatively simple.

However, the data rate measurement for direct streaming workflows is much more complex. It’s not sufficient to simply measure average data rates and look to maximize them. Streaming data on the fly to the printer requires a data rate measurement to be made at the page level. For instance, if you have a job that has 1000 easy-to-RIP pages but have one page in the middle that is very difficult (taking multiple seconds to RIP) then the average data rate may indicate you can keep up with the printer. When you are streaming the data and the pages must be delivered in order, the complex page in the middle of the job could force an underrun and starve the printer of data. It is no longer the complexity of the job but rather the complexity of individual pages that matters. This can be minimized by using elastic buffers to allow for some slow pages to be handled, but there is a limit to which this technique can be used and it will consume more memory.

The complexity of every page needs to be considered, which is determined largely by the ratio of complex vector content requiring compute time to turn it into a raster versus what is already raster data in the PDF. But, if the raster data in the PDF does not match the target device, you may still need to work on each pixel in the raster to convert into another raster.

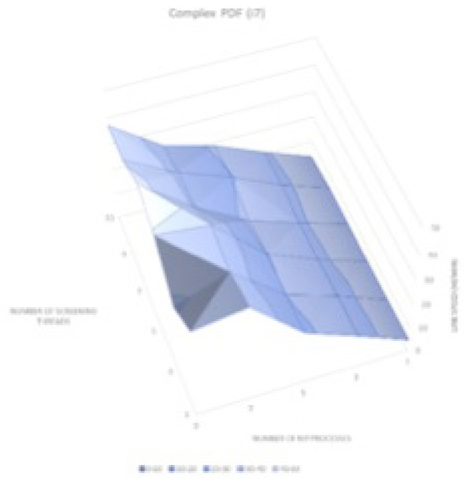
In the end, what is being sent to the printer is screened raster data but the PDF can contain varying mixes of complex vector mathematics and raw memory processing raster tasks. If your PDF has a high percentage of complex vector information, then having multiple powerful high clock speed CPU cores may be important when you are choosing PC hardware. If your PDFs have a lot of raster data, then very fast memory may be important. You may also have performance bottlenecks that appear as data moves from vector to raster. It is sometimes attractive to suggest that maximizing page data rates is just a case of adding faster cores, a GPU or faster memory, but this approach is too simplistic.

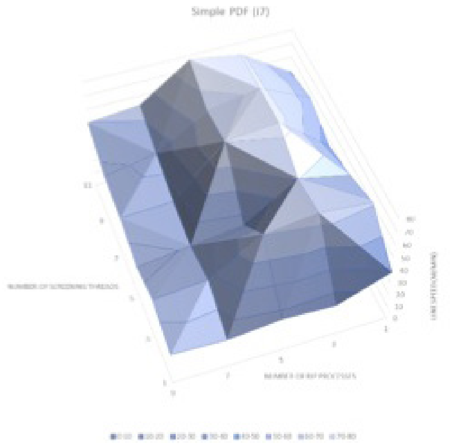


In this paper, we will discuss the complexities of selecting the software and hardware components at the heart of a workflow designed to process PDF jobs in real-time to stream data directly to the printhead electronics. We will show why we felt it was important to develop a dedicated tool that can measure and build models based on target PDFs, PC hardware and software (RIP) configurations in a more scientific way and enable machine learning algorithms to identify the critical parameters.

## Complexity of Job

As we move forward into the mass customization era, most digital print workflows will be driven by page description languages (PDLs). The most common PDL used to drive digital inkjet printers today is PDF (ISO-32000). Building a page using a combination of vector and raster elements can be a very efficient way to communicate information, but this can come at the cost of unpredictable print speeds. Two pages can look identical both before and after printing but the objects that create the page can vary hugely in complexity and therefore RIPping time. Once a page is RIPped to continuous tone, it then needs to be halftone screened. The halftone screening performance is dependent on the size of the images produced by the RIP and not the complexity of the objects in the original PDF.





The charts above show two different PDFs run through a RIP and halftone screener on a low powered "Microsoft Surface 4" laptop. The experiment measured the theoretical average press speed for a roll fed press printing a 13" wide media with a single colorant and a 2-bit screen. It then varied the number of processes given to the RIP (pages in parallel) and the number of threads given to the halftone screener. There are two PDF files shown above: a PDF with relatively complex objects and another with very simple PDF objects.

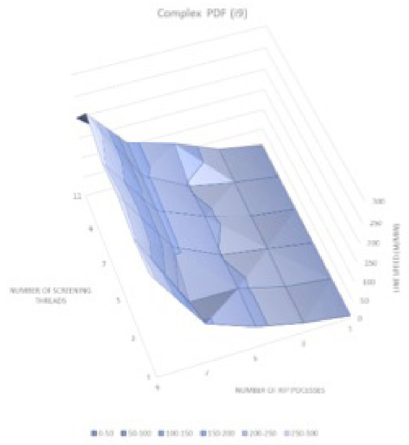
The results show both what you would expect and areas that require more experimentation. For instance, the complex PDF scales with the number of RIP processes. Adding more screening threads does not impact the line speed significantly. The RIPs are working hard to get raster data to the halftone screening thread which is then processing it very quickly. Adding additional screening threads just leads to screening threads waiting for the RIPs.

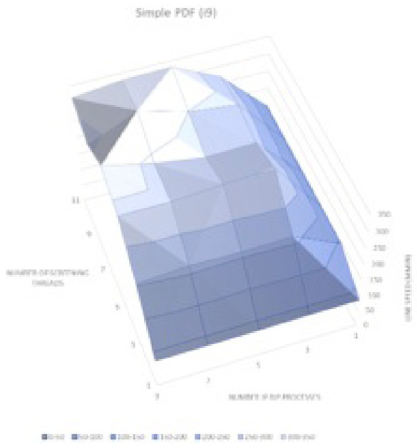
However, in the simple PDF the observation is reversed. The RIPs can get through the pages very quickly and the rasters start to stack up. The work that the screener threads are doing is a lot more than the RIPs for this PDF. Adding more screener threads allows the screeners to keep up with the RIPs leading to higher line speeds.

**Capability of PC Hardware**

Looking at the simple job test results further, we can see that the overall system slows down when there are more than three RIP processes. Monitoring the memory usage during the test gives us some insight into what may be happening. It appears that the RAM was not large enough to deal with the amount of data being created. When the RAM was full, the additional RIP processes hurt print performance.

If you run the same test on a much more powerful PC, then you get the results shown below. This PC was based on an Intel® Core™ i9-9980XE Extreme Edition Processor.





The first thing to note is the overall line speeds are much higher as you would expect with a faster CPU. It also has faster memory and a lot more of it. Again, the complex PDF is RIP process bound with very little impact from having extra screening threads.

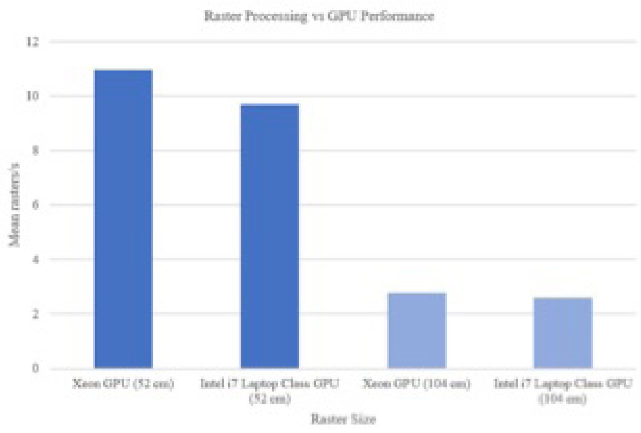
However, as the number of RIPs increases for the simple PDF, it does not show the slowdown in line speed. This could be the impact of having more and faster RAM. If the PDF had more image data than vector would fast memory be more important than clock speed? If the job had more complex vector data and transparency would the clock speed be the most important PC capability? What would the impact of adding in a GPU to add massive parallelization into the system (many hundreds of cores)?

## GPU Acceleration

In this experiment we looked at processing raster rotation and the screening on a GPU. There were three processes that happened for each image:

1) The image was copied from main memory to GPU memory.
2) The image was then rotated and screened on the GPU.
3) The image data was copied back into the main memory.

This test was initially done using a Laptop class GPU and then repeated with a Server class GPU (Nvidia Quadro P4000 on a Dell PowerEdge T430).



Although the GPU was very fast processing the data once it was in the GPU memory, there was an overhead copying the data on and off the GPU. This overhead was to some extent fixed and not dependent on the power of the GPU. Simply adding a more powerful GPU to the system did not speed things up significantly, as can be seen by moving from the laptop GPU to the server class GPU.

Using the Intel integrated GPU gave much faster transfer rates but the GPU was slower. The overall processing time was similar to the Nvidia GPU.

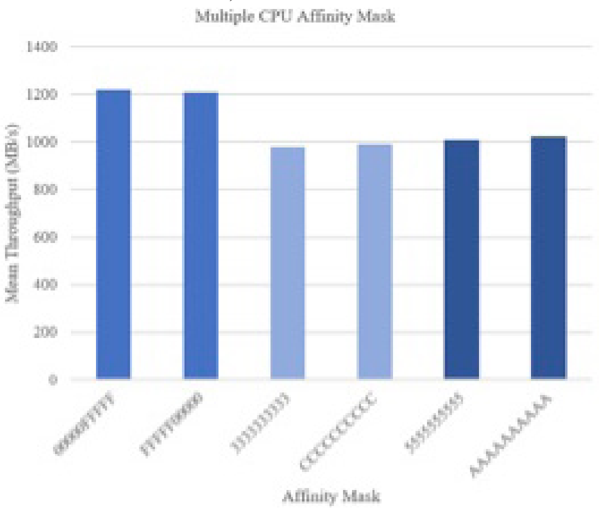
The experiment showed the opposite to what we expected to see. We expected to see a more powerful GPU giving significant multiples of performance. What we found was the overheads of moving data to and from the GPU memory was a bottleneck. This became more significant as the amount of data to be moved increased (i.e. with the bigger images).

## Multiple CPU

If you decided to keep all your processing on the main CPU, you may be tempted to add more cores by using multiple CPUs. Unfortunately, that increase in core count has a limitation. If you add in two CPUs then you have two memory pools (one per CPU). If the RIP thread is on one CPU and the screening thread on the other CPU, then the data access will be impacted. This may not be a problem if the information stored in memory is relatively small but if you need to access large high-resolution rasters this can result in a noticeable performance impact.

The results below show a dual CPU Intel Xeon with 10 physical cores per CPU with hyperthreading enabled (2 logical cores per physical core). The test application consisted of a RIP with 4 farm RIPs, 4 shared memory receiver threads, 7 screening threads, 2 packing threads and 1 output thread.

We used a feature called "Affinity mask" to force the operating system to run threads on each CPU in turn (00000FFFFF and FFFFF00000) and then across the two CPUs with multiple configurations (3333333333, CCCCCCCCCC and 5555555555, AAAAAAAAAA).



When the same number of threads are shared across the CPUs, an approximately 16-20% reduction in data rate can be seen. If the operating system could schedule the threads, it's likely that you would see this 20% drop in performance even though you have in theory more cores available.

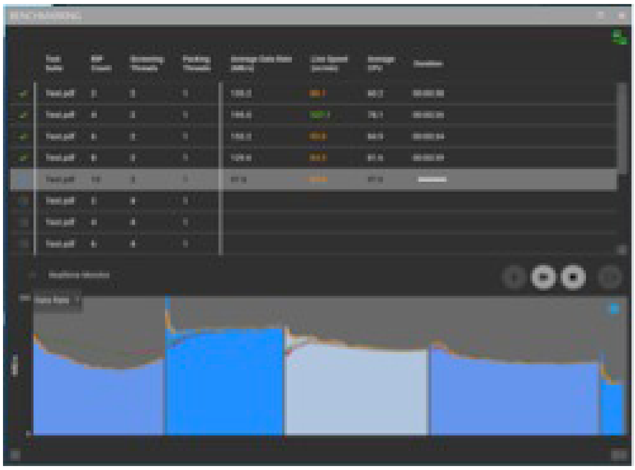
## Software (RIP) Settings

There are many performance settings in the RIP that can make classes of PDF jobs faster. They do this by selecting algorithms that are more efficient for certain PDF features. However, they must be turned on at the right times and may be slower if used at the wrong times. One general setting that can help is to rasterize and cache repeated vector objects so that they do not need to be re-RIPped every time.

However, there is a trade off depending on the capabilities of your hardware and the resolution of your output. If you have very quick powerful CPUs and fast clock speeds and output at a high resolution it may be quicker to RIP the objects repeatedly than incurring the overhead of handling the cached rasters in memory. If you have lots of very fast memory but less powerful cores, then caching objects can lead to a huge gain in performance.

## Developing a Benchmark Tool

The examples we have described in this paper show how PDF complexity, PC performance and RIP configuration combine to result in a specific data rate. When streaming directly to the printer at full print speed we explained that you need to measure data rates at the page level. The data rate we can maintain without starving the print electronics of data at any point defines the overall print speed.



We understood we needed to be able to rapidly run repeatable experiments and collect data in a controlled way. We have built a tool called Direct Benchmark which can run experiments on different PCs looking at combinations of PDFs, software configuration and PC capabilities. We run software that allows us to benchmark different components in the system and collect the data.

Direct Benchmark has been connected to an Azure cloud data store which allows us to see all the data in one place. We use artificial intelligence (Artemis) to learn from the data and create models that can be downloaded into the software.

These models can be used to intelligently configure the software so that we can make two levels of predictions: firstly, the achievable line speed given a particular PDF complexity and PC capability; secondly, the required PC capability and hence cost for a bill of materials for a given market. Using this model, we can predict the line speed of any PDF job that is to be sent to that printer. These individual models will be studied using big data and machine learning to develop generalized models that can predict what PC hardware is required for a new printer based on customer PDFs.