A Distributed Low-Cost RIP for Digital Presses

John Recker, Eric Hoarau, Wei Koh, I-Jong Lin; Hewlett Packard Laboratories, Palo Alto, CA.

Abstract

As commercial printing presses become faster, cheaper and more efficient, so too must the RIPs that process and feed them data to print. Previously [1], we presented a single machine GPU-accelerated RIP system that harnesses the massive parallel computational power of programmable Graphic Processing Units (GPUs). This project builds on this work and leverages the Apache Hadoop framework to construct a distributed RIP system designed to scale efficiently and meet current and future RIP requirements for high speed commercial digital presses. Hadoop is an open source package already deployed in mission critical application at major internet companies designed for massively distributed processing of large static data sets. It provides job management, load balancing, and error recovery services which are well suited to building a stable and comprehensive distributed RIP system. We found, however, that the Hadoop processing and data management models do not match the streaming media processing nature of an EPID ("Every Page is Different"[2]*) printing infrastructure. We have therefore implemented custom processing and data management modules to support distributed ripping of jobs and dynamic streaming of PDFs, images and ripped jobs through the system. Lastly, a custom front end was developed which allowed the system to be integrated with a digital press. The result is a solution combining open source software and a cluster of GPU-equipped commodity PC workstations which significantly reduces the cost and energy consumption of the DFE while scaling to meet the DFE needs of a wide range of commercial digital presses.*

EPID: the brave new PSP world

The business environment of the traditional Print Service Provider (PSP) is changing at a rapid rate, driven by the demands of customers increasingly accustomed to personalized web experiences, new business models such as Web-to-Print, and digital print infrastructure providers for which accelerating the transition from analog to digital printing is a key element of their growth strategies. Increasingly, print jobs consist of smaller runs targeted at narrow customer segments, or individual events. Web to print sites such as MagCloud.com [3] – a digital printing service which enables aspiring writers, artists and designers to create their own magazines, sell them on the web, then have them professionally printed and delivered - are examples of businesses that are greatly expanding access to PSPs to broader, though less sophisticated customer bases.

Where once "Variable Data Printing" (VDP) – a technique for sourcing portions of individual document from a database to provide limited personalization - was thought to provide the answer to customizing printed documents, it now seems that nothing short of a full "Every Page is Different" (EPID) – a workflow based on the assumption that every printed page is different from the last - will ultimately meet the needs of this new printing world. Achieving a true EPID workflow, however, requires re-evaluating every aspect of the print eco-system, from the business assumptions and models required to generate and deliver print jobs to the complete technical printing workflow: content creation, order submission, print production and delivery.

The internet infrastructure provides one model for this brave new world of democratized printing. From a technical perspective, the Internet is a vast collection of heterogeneous components connected by a stack of common protocols. Generalized frameworks with customized plug-able components have evolved for content endpoints (web browsers) and content sources (web server frameworks such as the Apache web server [4]). These customizable frameworks allow both continued backwards-compatibility with older content and protocols as well as supporting, if not encouraging, the evolution of the Internet. Internet business models have evolved that mirror this customized plug-in technical strategy. Both Amazon.com and eBay, for example, have built substantial businesses by encouraging third-party businesses to leverage the capabilities of their websites to sell their wares.

In general, it is our team's belief that the frameworks that successfully support the enormous range of Internet applications available today and have proven capable of reliably managing the high volume traffic typical of large web sites will ultimately prove better suited to supporting a web-driven EPID print infrastructure than the home-grown proprietary solutions typical of the print industry today.

RIP for an EPID print infrastructure

A flexible printing infrastructure requires a flexible, efficient RIP – one capable of matching both the print speeds of a wide variety of digital presses as well as the workflow flexibility required by a Print Service Provider (PSP) with single or multiple digital presses. Moreover, the RIP technology must scale to meet a variety of customer and press requirements. Digital print jobs are processed at widely variable rates, from under 10 to over 1000 pages per minute. The RIPs currently deployed to solve some of the more computationally demanding customer scenarios require large and expensive array of processors, and the cost of such systems can exceed $250,000. Therefore, a RIP solution is needed that scales from one to many machines, and is capable of supporting both ripping and printing at press speed. What is used on the low end (one machine)?

There are a variety of possible approaches to building a scalable, distributed RIP system. Our selection process was driven by a few requirements. In keeping with our philosophy regarding print workflows, we wanted an open source general purpose compute infrastructure capable of parallel execution of a variety of workloads on heterogeneous computing platforms. Furthermore, it had to be of sufficient quality to support a business critical

application, yet simple enough to program to allow us to meet an aggressive development schedule.

To this end, we adopted Hadoop MapReduce [5] - an Apache open source package for distributed processing of large data sets – as the framework on which to implement our distributed PDF RIP system. Hadoop MapReduce is attractive for several reasons. First, though an open source project, Hadoop is already deployed in mission critical application at companies such as Amazon, Yahoo! and Facebook. In 2008, for example, Yahoo! internally deployed a ~20,000 node Hadoop cluster running "hundreds of thousands of jobs" per month. Next it features a modular architecture with clear inter-module interfaces, designed to allow substitution or modification of key components for customization without compromising the integrity of the overall system.

The Hadoop MapReduce system

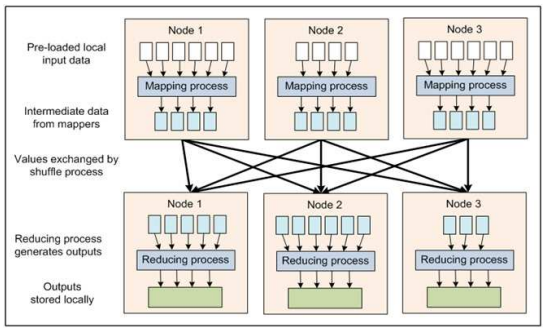

***Figure 1**. Hadoop MapReduce Architecture*

Although a full description of the Hadoop MapReduce system is beyond the scope of this paper, a brief summary of key features relevant to our work is presented.

Hadoop MapReduce is a Java open source software framework implementing the Map-Reduce [6] programming model for writing applications that rapidly process vast amounts of data in parallel on large clusters of compute nodes. In this programming model, large data sets are processed by a two stage map and reduce pipeline. Users specify a map function that processes an input data set to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key.

The Hadoop MapReduce framework, as shown in figure 1, is designed to massively parallelize the two stage Map-Reduce data filtering model. Programs written in this functional style are automatically parallelized and executed by the system on a large cluster of heterogeneous machines. Central to the implementation is a job submission and management system which provides services for distributed job management, load balancing, error recovery and status logging.

Hadoop also provides a distributed file system (HDFS) for automatically distributing and partitioning data across the processing cluster. Hadoop, in general, is optimized to run many jobs on extremely large but slowly changing data sets spread across thousands of computers. In this paradigm, it is easier to move the computation to the data than the data to the computation. Furthermore, to maximize fault tolerance, data is typically replicated on several machines.

The general usage pattern for the Hadoop MapReduce framework is as follows. A user defines separate map and reduce functions tailored to their needs either as Java classes or system command line invoked applications. Data is entered into the system using the Hadoop HDFS utilities. Jobs are then submitted to the Hadoop job management system, which automatically invokes parallel instances of the user specified map and reduce functions on the processing cluster to process subsets of the data in the HDFS. Once the job is completed, the user retrieves job results from the HDFS.

A Hadoop MapReduce based distributed RIP

While many of the features of the Hadoop system are compelling – automated distributed job management, fault tolerance, proven performance and reliability – neither the Map Reduce programming model, nor the Hadoop distributed data management model are suitable as–is to a distributed RIP system.

In particular, RIP requires a single processing stage and not a dual map-reduce process. Furthermore, the data processing requirements of an EPID printing infrastructure are closer to that of a streaming media system than the traditional batch processing Digital Front End (DFE) system architectures common today. In this model, Print Description Language (PDL) files such as Adobe PDF [7] are streamed through a RIP filter which transforms device independent language page specifications into device dependant pixels, which, in turn, are then streamed to the digital press. Like a typical streaming media system, EPID RIP systems face real time constraints: roll-to-roll presses cannot stop and the data generators must have data ready for the print heads at all times. In fact, if the RIP sub-system is truly always able to transform PDL files at print speed, then one can envision removing the disk farms typically used to buffer pre-ripped image files common to today's press DFEs – a significant contributor to DFE hardware costs.

As a result, we needed to customize both Hadoop's processing model and implement an alternate data management system for moving PDFs, images and ripped jobs through the system.

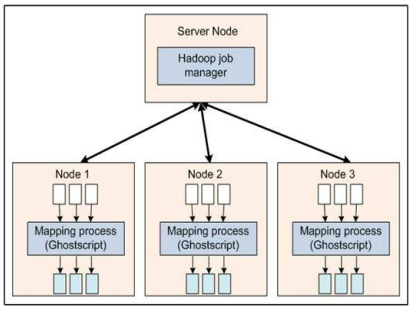

***Figure 2**. Hadoop RIP Architecture*

Our solution, as shown in figure 2, utilizes the Hadoop streaming interface which allows the Hadoop job manager to directly invoke non-Java applications. Our application, a combination of Python and Bourne shell scripts and "C" executables, is invoked as a first stage Map function by the Hadoop Job processor. It receives the name of a PDF file to be ripped from the framework, copies the PDF file from a central server to a cluster RIP processor, invokes an instance of our GPU-RIP accelerated Ghostscript interpreter for the PostScript language

and for PDF [2][8] based RIP, then copies the resulting images back to the server. Data copying is performed to and from a central server-resident drive mounted on each of the rendering clients. Since only one processing stage is needed, the system is configured to not invoke a Reduce processor.

ASCII configuration files on the central Hadoop control node manage which machines are available to the RIP cluster, and how many RIP instances (as map functions) to issue per machine. A custom Python front-end was designed for our hardware and software environment. This front-end submits collections of PDF files to be ripped as independent Hadoop jobs.

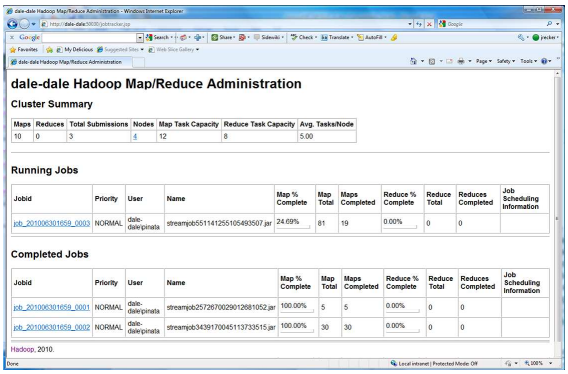

***Figure 3**. Hadoop Job Tracker web browser interface*

The RIP jobs are queued by the Hadoop job manager, and automatically issued to idle RIP instances on the cluster. Figure 3 shows the web browser interface to the Hadoop job manager. In this example, the cluster summary shows there are four rendering nodes, each with a maximum of three RIP instances (map instances in the table). The job manager in this examples further shows two completed Hadoop jobs and one active job composed of 81 PDF documents (one map is issued per PDF document).

The target job mix for this project consists of relatively low page count PDF color marketing collateral documents. As a result, each Hadoop task rips an entire PDF document and parallelism is supported on a per-document basis. As Ghostscript supports an option for specifying document page ranges to be ripped, it would be straightforward to extend the system to parallel ripping of single documents at the expense of copying the input PDF documents to the render cluster once for each parallel RIP instance.

Results

A PC cluster running our integrated Hadoop /Ghostscript/GPU-RIP solution was assembled in mid 2009. The project was initially assembled by first time Hadoop users in roughly four months. The cluster is centered around a HP Proliant ML370GS server running Windows, which acts both as the central Hadoop control node, and distributor of ripped jobs to the press. Ripping is performed on four Ubuntu Linux HP z800 single quad-cores Intel Xeon processor workstations with NVidia GTX285 GPUs, 3Gbytes of memory, and each connected via separate 1 Gb Ethernet network links to an 4 ports network card on the server. Rendered jobs in the form of compressed contone or 1-bit per channel Tiff images are output to a shared directory on the server. This directory is, in turn, monitored by the press control sub-system which forwards the ripped results directly to the press for printing.

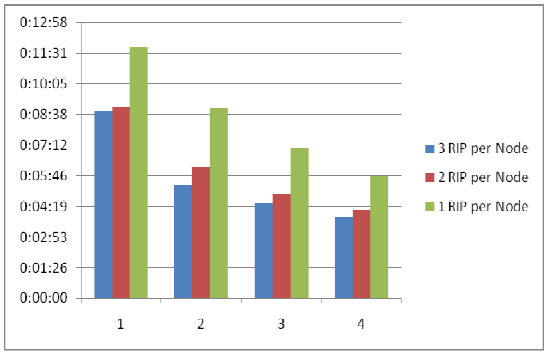

***Figure 4**. Hadoop RIP Performance scaling. Horizontal axis is number of rendering nodes; vertical axis is total rendering time.*

Figure 4 plots the total rendering time for a job consisting of a collection of marketing brochures and documents with a total of roughly 7500 pages. Total rendering time measures the time interval between job submission to the Hadoop job manager and the rasterized documents being copied back to the central server; including the time required to distribute the document source to the rendering nodes, the time to rasterize the documents, and any Hadoop job manager overhead. The documents range in length from between ten and fifty pages, and many of the documents feature color imagery. This collection was selected as representative of a typical mix of documents printed on digital presses.

Performance scaling, as shown in figure 4, varies as a function of the number of machines and RIP instances per machine. In general, the performance scaling is quite linear as a function of the number of machines for a single RIP instance per rendering nodes. However, performance scaling flattens out as a function of the number of rendering nodes as the number of RIP instances increases per machine. For example, increasing the number of rendering nodes from three to four does not substantially increase the aggregate RIP throughput when executing two RIP instances per machine. This behavior is particularly noticeable when executing three RIP instances per machine. In this case, the aggregate rendering performance is not substantially better than executing two RIP instances per rendering nodes. Likely this result is mostly due to system memory limitations in the rendering nodes. Previous experiments with a single machine have shown that multiple RIP rendering performance increased significantly with additional system memory.

Conclusion

The transition from analog to digital printing offers both the opportunity and challenge of re-examining every aspect of the printing business. On the opportunity front, an EPID workflow allows entirely new business models and opens commercial printing to new customers segments for which traditional high-volume analog presses where inappropriate. These opportunities however are not without risk. Not all business models will

succeed and not all customer segments will adopt these new technologies. It is our belief that leveraging proven technologies from other parts of the IT world will, in the end, provide the easiest, lowest cost, and most robust means of facilitating this transition.

Re-purposing the Hadoop framework for our distributed RIP system is an example of this approach. Not only did the system we developed provide the performance needed for our project, but it seems likely that, at best, we could only have developed a rudimentary custom framework in the time it took to develop this solution. On the other hand, Hadoop is far from a rudimentary framework. Not only has it already proven itself capable of supporting business critical applications, the Hadoop framework is supported by an active community of developers. For example, while we were building our system, another unrelated team of developers was also exploring the difficulties of streaming data through the Hadoop HDFS file system and developed a different solution to the problem [9]. Likely, this is not the final word on this problem. Instead, the problem of adapting Hadoop to better handle streaming data processing seems like a rich area for research contributions as much of the work in streaming media systems seems concentrated on network protocols and less on computational frameworks.

Similarly, the results presented here suggest to us that there are likely other significant opportunities for leveraging software frameworks developed for other domains to digital commercial press workflows and further accelerating the transition to digital printing.

References

[1] J. Recker, G. Beretta, I-J. Lin, "Font rendering on a GPU-based raster image processor", Proc. SPIE, Vol. 7528 (2010).

[2] K. Moore, "Every Page is Different: A New Document Type for Commercial Printing", ACM Doc. Eng. (2006).

[3] Kok-Wei Koh and Ehud Chatow, "MagCloud: magazine self-publishing for the long tail", Proc. SPIE, Vol. 7540 (2010).

[4] http://httpd.apache.org/

[5] http://hadoop.apache.org/

[6] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA. (2004).

[7] Adobe Systems Incorporated, PDF Reference, 6th edition, Adobe® Portable Document Format, Version 1.7 (2006)

[8] http://www.ghostscript.com/

[9] Russell Perry, "High Speed Raster Image Streaming For Digital Presses Using the Hadoop File System", HP Labs Tech. Report, HPL-2009-345, (2009)

Author Biography

John Recker is a researcher with the Print Production Automation Laboratories at Hewlett-Packard Laboratories in Palo Alto, CA. He has spent a now almost frightening number of years building graphics systems for window systems, CAD and gaming in both commercial and research environments, and now is focused on bringing the benefits of GPUs and parallel systems to digital print production for commercial printing. John received a BSEE from Cornell University.

Eric Hoarau is a senior researcher with Hewlett-Packard Laboratories (Palo Alto, California). He received his B.S. from the University of California at Berkeley and his M.S. in Mechanical Engineering from the Massachusetts Institute of Technology. His research interests span several fields: Mechatronic systems, imaging systems, color imaging algorithms, distributed computing and system dynamics.

Wei Koh is a color imaging researcher in the Print Production Automation Lab at Hewlett-Packard Laboratories in Palo Alto, CA. He earned his B.S. in Computer Science from the University of Washington and his M.S. in Computer Science from Stanford University. He is currently serving as associate editor for the Journal of Imaging Science and Technology (JIST).

I-Jong Lin is currently a Principal Research Scientist at the Print Production Automation Laboratory at Hewlett Packard Labs, Palo Alto, CA, USA. He received his Ph.D. in Electrical Engineering from Princeton University, and his M.S. and B.S. from Stanford University.