Mitigating Challenges of Cloud “Bursting”

spillbox

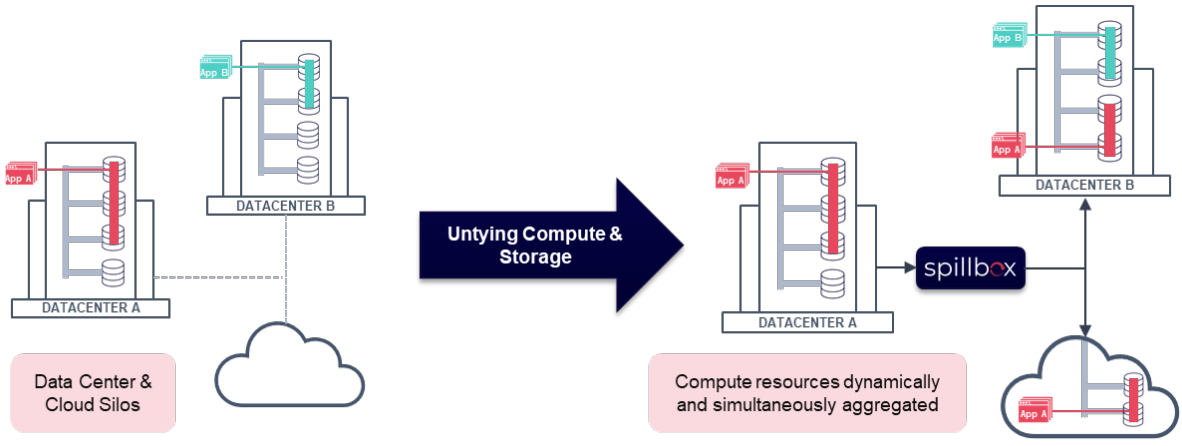
info@spillbox.io | spillbox.io

Introduction

Despite the stellar rise of modern Cloud computing since the early 2000's, there are still applications and entire industry segments that primarily rely on traditional On-Premise/Private Datacenter computing. Industries requiring high compute resources such as EDA and semiconductor development are facing challenges in utilizing advantages of the Cloud:

- Difficulty of implementing Cloud "bursting" and maintaining consistency of data between On-Premise and the Cloud
- Lower performance in running NFS based traditional workloads on scaled-out Elastic File System in the Cloud
- Additional storage, and associated cost, needed to move data to the Cloud
- Security exposure with putting libraries, IP and design on the Cloud
- Dependency on a particular Cloud provider, which results in lock-in and hence higher cost due to lack of competition

Spillbox recognized that the essence of the problem is a tie between compute and data/storage and that by removing this co-location dependency, the compute can dynamically scale by utilizing resources in the Cloud or an underutilized datacenter. The core of the Spillbox solution is the NFS-based network file system that allows the majority of data (>90%) to remain at the source while the compute moves to the Cloud. The entire process of dynamically acquiring compute resources in the Cloud and moving only the necessary data is automated, making it extremely easy to manage workflows.



At Spillbox, we have been working with semiconductor and EDA companies to prove the feasibility of our solution. This paper highlights the experiments that we completed and the results from those experiments with one company (Customer X).

Design Setup

Two designs were used for this experiment with the focus on physical design. One is the Bitcoin design, which is a low power ASIC design and the second is the NVDLA, which is Nvidia Deep Learning Accelerator design.

Prior to our experiment, Customer X had ported Bitcoin design to the AWS Cloud (without Spillbox), obtained performance results and quantified additional effort and storage required to do the porting for that design. In contrast, Customer X was porting the NVDLA design to the Cloud for the first time. These two designs give a good indication of some of the ways in which customers can benefit using Spillbox technology.

©Spillbox.io

Tools & Testbed Infrastructure

Industry standard tools from Synopsys are used for this experiment. These are:

1. Design Compiler (2016.12-SP2)
2. IC Compiler II (2016.12-SP2)
3. Formality (2016.12-SP2)
4. Primetime (2016.12-SP2)

The table below represents the testbed infrastructure which shows the different File Systems used and also CPU cores per socket, which were 18 for on-premise and 16 for AWS Cloud with and without Spillbox solution.

	On-Prem	Manual on AWS (without Spillbox)	Spillbox on AWS
AWS Instance Type	N/A	r4.16xlarge	r4.16xlarge
CPU – vCPUs	36	64	64
CPU – threads per core	1	2	2
CPU – cores per socket	18	16	16
CPU - sockets	2	2	2
CPU – model	E5-2697 v4 @ 2.30GHz	E5-2686 v4 @ 2.30 GHz	E5-2686 v4 @ 2.30 GHz
CPU – L1d	32K	32K	32K
CPU – L1i	32K	32K	32K
CPU – L2	256K	256K	256K
CPU – L3	46080K	46080K	46080K
Memory – online	252G	488G	488G
OS	CentOS 6.6	CentOS 7.5	CentOS 7.6
Filesystem	**NFS (Tier 2)**	**EFS**	**SPLFS***

Experiment 1: Bitcoin Design

The Bitcoin microarchitecture is shown below.

1. BIT_SLICE is the fundamental block and consists of SIPO (Serial In-Parallel Out) block, 2 memories and PISO (Parallel In-Serial Out) block
2. BIT_TOP includes 32 BIT_SLICE instances, in-addition to SIPO and PISO blocks
3. BIT_COIN includes 16 BIT_TOP instances, along with PISO and HASH blocks

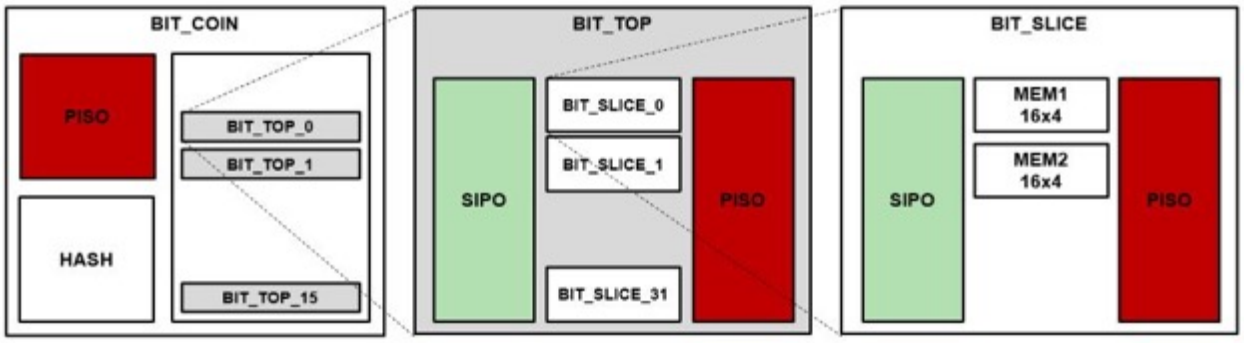


Figure 2.1 - Bitcoin Microarchitecture

Details related to the size of the Bitcoin design are shown below:

Design Type	Low Power ASIC
Technology Node	32nm
Technology Libraries	Synopsys SAED32
Total Number of Nets	174.5K
Total Number of Pins	1.02M
Hierarchical Cell Count	5.7K
Hierarchical Port Count	163.0K
Leaf Cell Count	244.8K

Performance Benchmarks

The table below summarizes the total time it took to run the complete backend physical design in three scenarios.

1. On-prem
2. On AWS Cloud
3. On AWS Cloud using Spillbox solution to run these EDA tools

Metrics		On-Prem (hrs)	Manual on AWS (hrs)	Spillbox on AWS (hrs)	% Delta VS AWS	% Delta VS On-Prem
Bitcoin Design	Elapsed Time	22.19	22.71	**20.50**	**-9.7%**	**-7.6%**
	User CPU	31.48	29.35	**28.68**	-2.3%	-8.9%
	System CPU	0.37	0.67	**0.65**	-2.1%	77.9%
	Total CPU	31.84	30.02	**29.33**	-2.3%	-7.9%
	CPU %	143%	132%	**143%**	+8.2%	0.0%
	Peak Memory	35.2 GB	35.5 GB	**35.2 GB**	-0.8%	0.0%
Spillbox.io	Total Data Transferred			**10.0 GB**		
	# of File Metadata Transferred			**84.5K**		
	# of Files Transferred			**3.7K**		
	# of Workers			**1**		

Initial concern of the Customer X was that there will be a significant degradation of performance with the Spillbox solution where the data (that remains on-premise) is accessed via WAN while compute is moved to the Cloud. However, based on the test results, we see 9.7% improvement when running on AWS Cloud with Spillbox (data on-premise) compared to running on AWS Cloud without Spillbox (data in the Cloud). The reason for this improvement is because patented technology used in Spillbox File Server (SPLFS) that better handles the NFS-based EDA workloads compared to Elastic File Server (EFS), which is optimized for scaled-out designs.

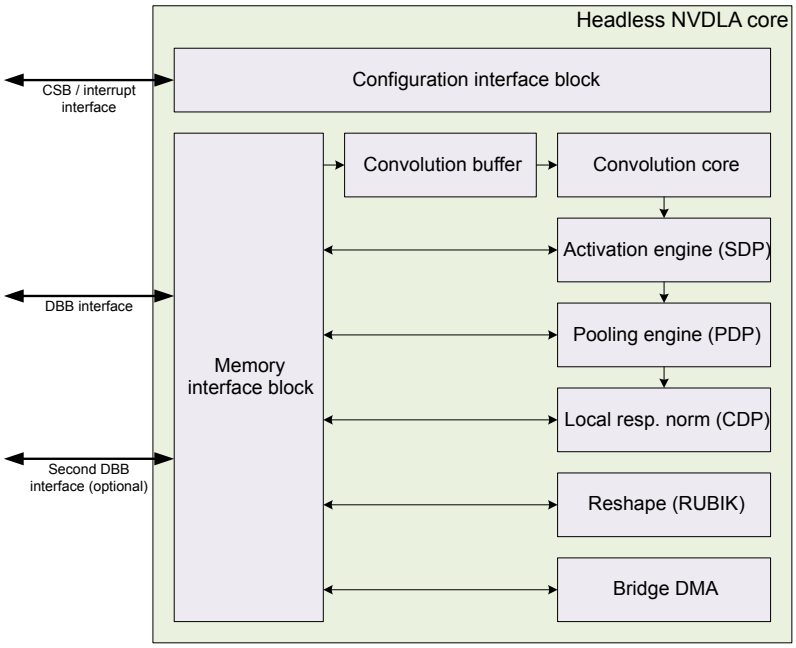
We also see a performance improvement of about 7.6% compared to the base case i.e. running the design on-prem. The primary reason for this improvement is because Spillbox FS can make use of Cloud's NVMe infrastructure to get better IOPs on the Cloud, compared to on-prem.

In addition to the compute run time, in AWS without Spillbox flow, user is responsible for moving required data to the Cloud. This is generally done manually using some scripts by DevOPs or syncing the entire filesystem which could be quite large and expensive.

©Spillbox.io

Experiment 2: Nvidia Deep Learning Accelerator (NVDLA) Design

Below is the architectural block diagram of the NVDLA core.



Details related to size of NVDLA design are shown below:

Technology Node	32nm
Technology Libraries	Synopsys SAED32
Block	NVDLA_partition_a
Total Number of Cells	1.13M
Clock	1.1GHz
FFs	408K

Performance comparisons show about 4.9% improvement to on-premise.

NVDLA design was not Cloud-ready and Customer X was porting this design for the first time. So, we looked at the metrics related to how easy it is to package it for the Cloud. Normally, moving the data to the Cloud is a manual process, but with Spillbox it is automated and only the data that is required for computation is moved by Spillbox to the Cloud.

Here is quotation from Customer X's designer, who worked on this.

"NVDLA was not Cloud-ready when I received it. The design's root directory contained a symbolic link to a separate partition housing the design's reference libraries. In addition to this, two scripts within NVDLA_partition_a were independently hard-coding another external path. I leveraged the logs from the completed Spillbox run to identify all externally referenced files. With that information, I was able to quickly create a self-contained, Cloud-ready package for the traditional AWS baseline run (sans Spillbox).

Without a tool like Spillbox, creating such a package requires either:

1. *A high-level of expertise and familiarity with all aspects of the design*
2. *A brute force approach – potentially uploading hundreds of GBs of unnecessary data*
3. *A painfully slow onion-peeling approach – uploading additional required files with each consecutive failed run*

I am all too familiar with both 2) and 3) from a previous project where I was tasked with packaging multiple designs for an AWS evaluation. One of those designs was "Bitcoin", which I would estimate consumed 15 manhours to debug/contain before an end-to-end AWS run came out clean."

This clearly indicates a reduction from 15 manhours to 1.5 manhours, which is a 90% improvement in the effort to run a design which is not Cloud ready. The Spillbox tool automatically identifies and pulls all the files required for a successful Cloud run.

We also looked at the actual size of the external reference paths and compared it with the size of the data that was needed from those paths. The table below shows this comparison:

External Reference Path	Actual	Required
/slowfs/gts_lowpower/saed_lp_training_libs	8.4G	58M
/slowfs/gts_lowpower/training/machine_learning	13G	1.8G
/slowfs/gts_lowpower/training/tech_file_testing	1.1M	84K
	22G	**1.9G**

Here the actual size of the external reference path is 22 GBytes, but what was needed to run the design was only 1.9 GBytes, which is 91% reduction. This shows another benefit of using Spillbox tool, which is reduced storage in the Cloud. This is again because rather than moving all the data manually to the Cloud, only the required data gets moved automatically by Spillbox.

Summary

This paper demonstrates how Spillbox can address some of the challenges associated with using the Cloud for semiconductor designs. The above two experiments done by Customer X using Spillbox solution shows three main benefits:

(a) Improved performance when running backend physical design tools
(b) Ease of migration and consistency of data between on-prem and Cloud
(c) Reduced storage on the Cloud when using Spillbox, as it copies only what is needed

Couple of other benefits with Spillbox solution are:

(d) User and/or IT can control what files need to be sent to the Cloud. This addresses some of the security concerns.
(e) User and the company are not bound by one Cloud provider, as it is quite easy to migrate. This provides flexibility to the customer and reduces their total cost of ownership.

These two experiments show the saving in time and storage on single design. Real build and regression in semiconductor development are significantly more complex. Spillbox mitigates those complexities and 9-12 months of DevOps Cloud migration is reduced to days.

In closing, I would like to add another quotation from Company X engineer

"Overall, my experience with Spillbox has been quite good. Your product makes it very easy to burst workloads to the Cloud by taking the hassle out of packing and uploading software, data, etc. I am confident that even users with no Cloud experience can be up and running within 15 minutes."

©Spillbox.io