Migrate and right-size HPC workloads on Microsoft Azure

Altair Breeze use case for the semiconductor industry

By Michael Requa, Senior Program Manager, Microsoft
Rosemary Francis, Chief Scientist, Altair

May 2021

Semiconductor companies are taking advantage of Microsoft Azure high-performance computing (HPC) infrastructures for their complex electronic design automation (EDA) software. When a client, a large semiconductor company, asked for help using Azure to run its EDA workload, Microsoft teamed up with Altair, a global technology company providing solutions in simulation, HPC, and AI.

The client was running a relatively large design of 100 million transistors. We used Altair Breeze™, an I/O profiling tool, to troubleshoot and tune the company's workload before and after migrating it to Azure. Breeze revealed I/O patterns that had previously gone unnoticed and showed us how to significantly improve application runtime.

This article, jointly written by the Microsoft and Altair team in charge of the migration, describes our journey. We show how we used Altair Breeze to diagnose I/O patterns, choose the workflow segments best suited for the cloud, and right-size the Azure infrastructure. The result was better performance and lower costs for our semiconductor customer.

Choosing the right workflow to migrate

The EDA workloads used for semiconductor design and verification require complex, segmented workflows and multiple, high-speed software tools. Only some of the workflows suit a cloud-burst pattern—that is, a configuration between a private cloud on-premises and a public cloud to which overflow traffic is directed. The semiconductor company needed help determining the part of its workflow to migrate to Azure.

To find out, we used Breeze to create a profile of the EDA workflow on-premises. As an application profiler, Breeze is like a system debugger that lists every file, library, application, and network dependency. That made it easy to see exactly which components we needed to migrate—and which were better left on-premises.

Without the type of information that Breeze provides, teams often spend a lot of time trying to discover application dependencies by trial and error. The alternative—moving *all* the data to the cloud—is very expensive. In this case, the power integrity and sign-off workload was a good choice to run on Azure.

It didn't need multiple terabytes of data movement and would benefit from the autoscaling services that Azure provides.

With just one run of the workflow, Breeze listed everything needed to run the selected EDA workload. Three dependencies stood out:

- Callisto, the code-name for the tool used in power-integrity testing.
- Skyline, the code-name for a segmentation utility used to operate the power-integrity workload efficiently.
- Python 2.7.13 libraries.

The next step was to set up the right Azure infrastructure for this workload.

Right-sizing an HPC infrastructure on Azure

Azure provides a collection of services for running EDA workloads. Which works best? We used Breeze to help make that determination.

The Breeze I/O summary from the on-premises analysis showed that Callisto was an I/O-intensive workload and a multithreaded CPU process. Callisto can also run in a scale-out format where portions of the work are divided across a cluster and other portions can use a local disk. Azure Virtual Machine Scale Sets suit this type of work and make it easy to create and manage multiple virtual machines.

Azure offers a wide variety of virtual machine sizes to support many types of uses. A price-performance analysis showed us the best option for Callisto from the HB-series, which is designed for HPC computing. We chose the HBv3-series virtual machines, which feature 120 cores of AMD EPYC (Milan) 7003 CPUs capable of 3.65 GHz and 2 × 1 TB attached SSD drives.

For the shared Network File System (NFS) storage, we chose Azure NetApp Files, a managed service that makes it easy to migrate and run complex, file-based applications. NetApp Files supported the peak shared file system load required by this workload—150k IOPS. Both Skyline and the Python libraries were installed on this shared NFS storage.

We also needed to handle a dependency on Altair® Grid Engine®, a distributed resource management system installed on-premises. Skyline and Python communicated with Grid Engine through a connector. Integration meant configuring Grid Engine for the new HBv3 scale set on Azure—and virtually no scripting changes were needed.

To orchestrate and manage the HPC environment on Azure, we used Azure CycleCloud. It has the advantage of simplicity and works with any scheduler. CycleCloud automates cluster creation and management, including Grid Engine clusters, so it was an easy choice.

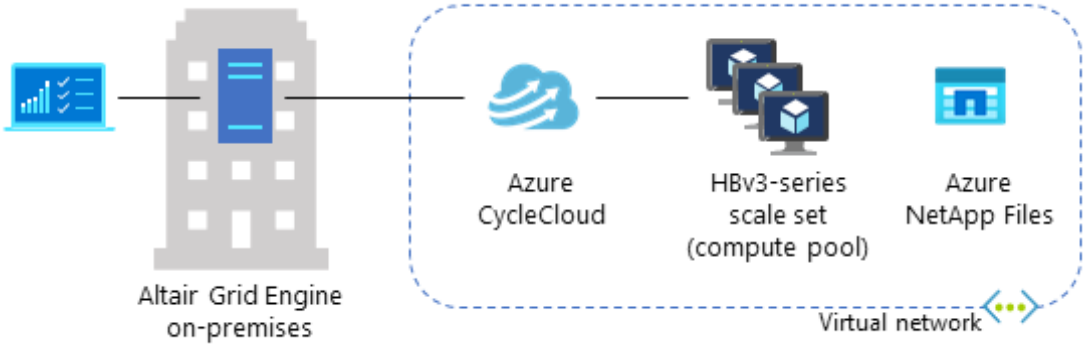
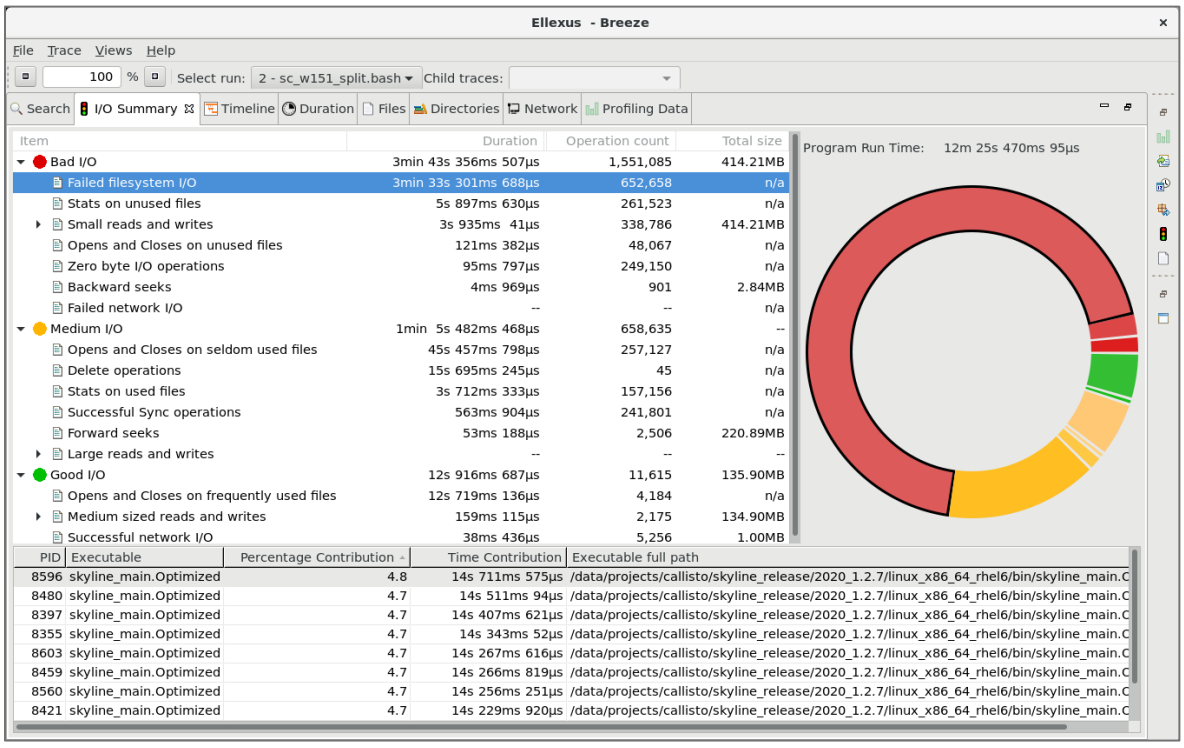


Figure 1. The Azure HPC infrastructure uses Azure CycleCloud to orchestrate the clusters. NetApp Files provides high-performance storage. Altair Grid Engine is configured to work with the compute pool on Azure.

Benchmarking the new Azure HPC infrastructure

Our next step was to make sure the EDA workload ran as well on Azure as it had on-premises. Again, we turned to Breeze for its ability to record I/O patterns and show how an application uses the network and file system. We got good news—the performance on Azure was similar to performance on-premises. However, Breeze highlighted a potential bottleneck.

The Breeze I/O summary shows the I/O breakdown for an application and classifies the results as good, medium, or bad, making it easy to see where problems may lie. As Figure 2 shows, Breeze highlighted 3 min, 43 sec spent making failed I/O calls over a runtime of 12 min, 25 sec. Breeze recorded 652,658 failed I/O calls in total—far more than we expected.



Item	Duration	Operation count	Total size
Bad I/O	3min 43s 356ms 507µs	1,551,085	414.21MB
Failed filesystem I/O	3min 33s 301ms 688µs	652,658	n/a
Stats on unused files	5s 897ms 630µs	261,523	n/a
Small reads and writes	3s 935ms 41µs	338,786	414.21MB
Opens and Closes on unused files	121ms 382µs	48,067	n/a
Zero byte I/O operations	95ms 797µs	249,150	n/a
Backward seeks	4ms 969µs	901	2.84MB
Failed network I/O	--	--	n/a
Medium I/O	1min 5s 482ms 468µs	658,635	--
Opens and Closes on seldom used files	45s 457ms 798µs	257,127	n/a
Delete operations	15s 695ms 245µs	45	n/a
Stats on used files	3s 712ms 333µs	157,156	n/a
Successful Sync operations	563ms 904µs	241,801	n/a
Forward seeks	53ms 188µs	2,506	220.89MB
Large reads and writes	--	--	--
Good I/O	12s 916ms 687µs	11,615	135.90MB
Opens and Closes on frequently used files	12s 719ms 136µs	4,184	n/a
Medium sized reads and writes	159ms 115µs	2,175	134.90MB
Successful network I/O	38ms 436µs	5,256	1.00MB

PID	Executable	Percentage Contribution	Time Contribution	Executable full path
8596	skyline_main.Optimized	4.8	14s 711ms 575µs	/data/projects/callisto/skyline_release/2020_1.2.7/linux_x86_64_rhel6/bin/skyline_main.C
8480	skyline_main.Optimized	4.7	14s 511ms 94µs	/data/projects/callisto/skyline_release/2020_1.2.7/linux_x86_64_rhel6/bin/skyline_main.C
8397	skyline_main.Optimized	4.7	14s 407ms 621µs	/data/projects/callisto/skyline_release/2020_1.2.7/linux_x86_64_rhel6/bin/skyline_main.C
8355	skyline_main.Optimized	4.7	14s 343ms 52µs	/data/projects/callisto/skyline_release/2020_1.2.7/linux_x86_64_rhel6/bin/skyline_main.C
8603	skyline_main.Optimized	4.7	14s 267ms 616µs	/data/projects/callisto/skyline_release/2020_1.2.7/linux_x86_64_rhel6/bin/skyline_main.C
8459	skyline_main.Optimized	4.7	14s 266ms 819µs	/data/projects/callisto/skyline_release/2020_1.2.7/linux_x86_64_rhel6/bin/skyline_main.C
8560	skyline_main.Optimized	4.7	14s 256ms 251µs	/data/projects/callisto/skyline_release/2020_1.2.7/linux_x86_64_rhel6/bin/skyline_main.C
8421	skyline_main.Optimized	4.7	14s 229ms 920µs	/data/projects/callisto/skyline_release/2020_1.2.7/linux_x86_64_rhel6/bin/skyline_main.C

Figure 2. Altair Breeze I/O summary showing the time spent in bad, medium, and good I/O.

We took a closer look using the Files view, which showed numerous failed stat()and open() calls. Stat() calls are made when checking the permissions and existence of a file. Applications often do this when looking for files that don't exist. For example, an application with a path variable continues to search many locations before trying to open the correct file.

We sorted by the number of failed stat() calls to see the paths that were most often checked. The file with the most stat() operations included a single path to a file called <string>, which was accessed 91,000 times! That told us that the application had a bug. However, those operations took only 263 ms in total, suggesting it might be noisy but insignificant.

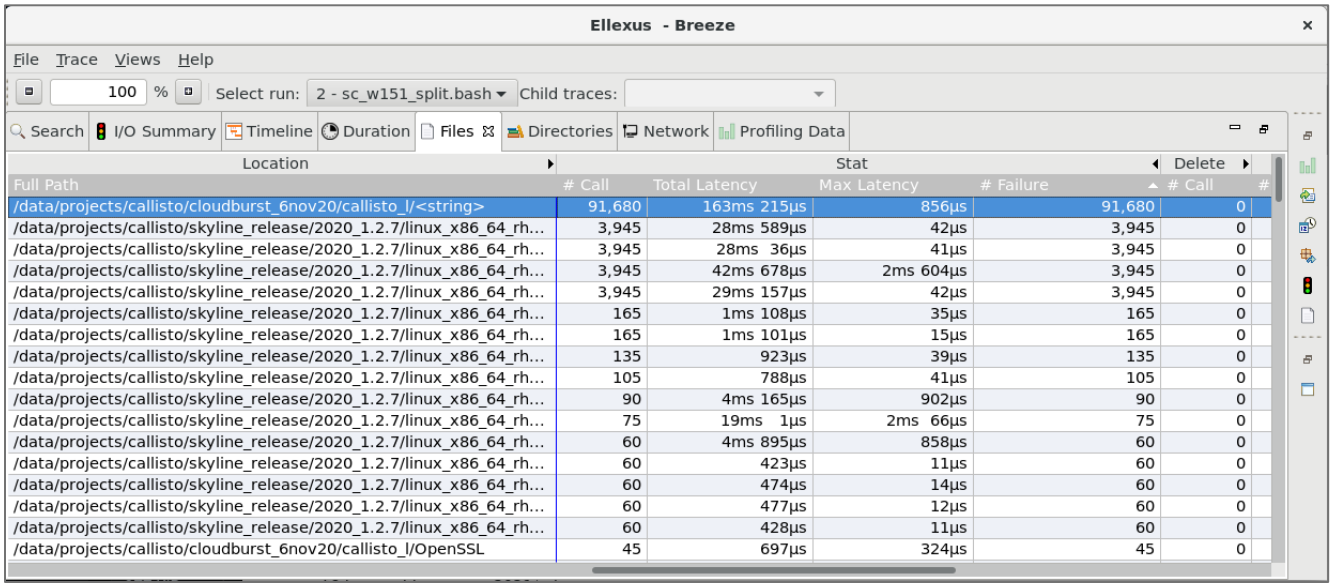


Figure 3. The Files view in Breeze shows every file used by an application and its I/O patterns.

In fact, we saw that most of the failed stat() and open() operations were spread over a wide number of file paths, each with only 15 to 60 operations apiece. The paths nearly always pointed to a Python library.

When we opened the Timeline view, we saw that the application is run in parallel with 15 separate processes. By selecting one of the worker processes and opening the related Events view, we could list every operation performed by that process. We saw that every process searched many different paths for each Python library. In other words, for each Python library used, dozens of operations failed before the correct path was tried.

Failed I/O is expensive. It's rarely cached, and it drains the IOPS budget of Azure NetApp Files. Some of the failed operations we saw took more than 1 ms, which soon adds up.

A Python module path is configurable. When this machine image was set up, it's likely that the Python installation was not properly configured and included unneeded paths. By removing those unused paths from the module path configuration, we were able to significantly speed up this workflow.

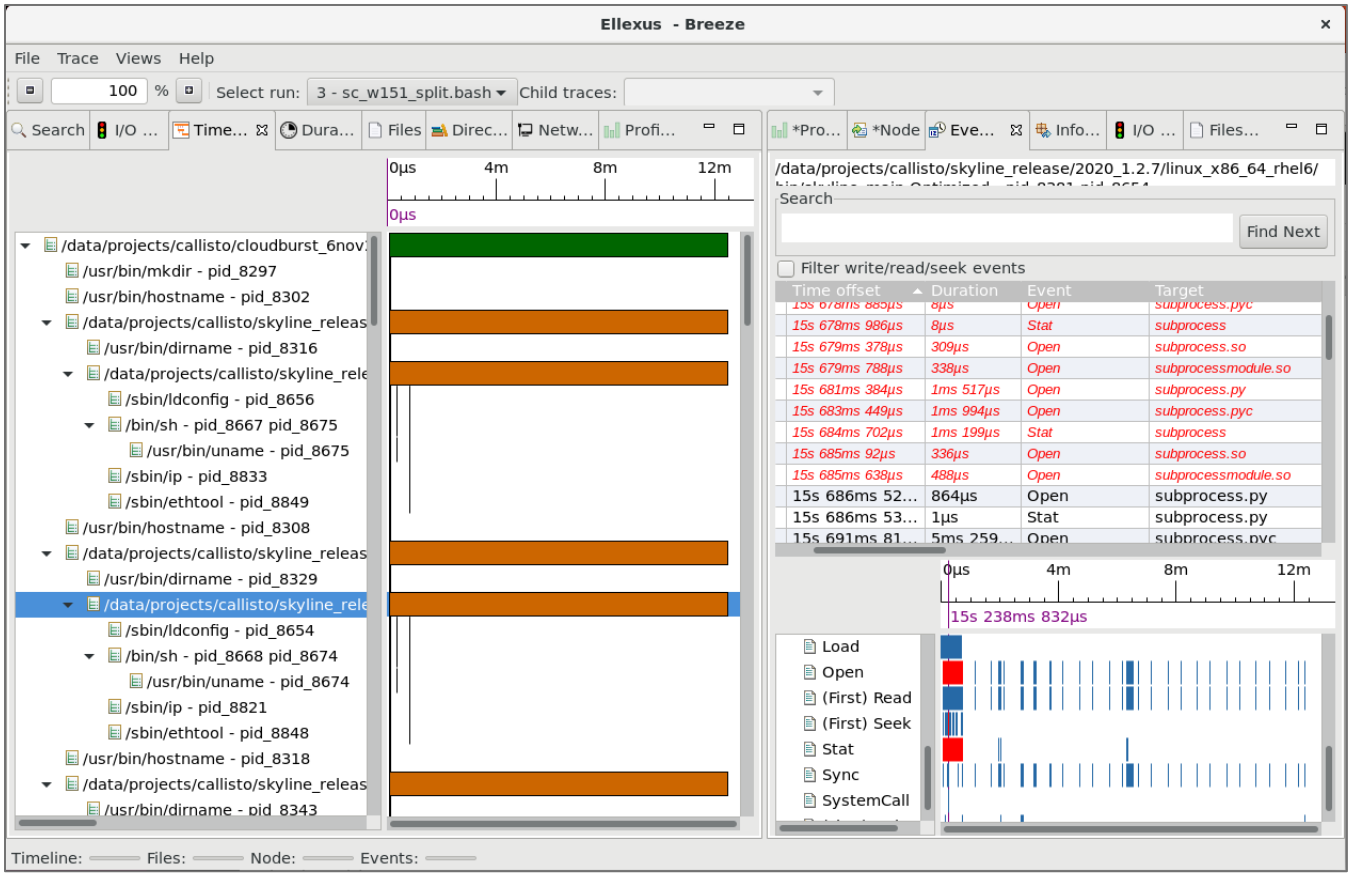


Figure 4. The Timeline view in Breeze shows the process tree, while the Events view shows every operation performed by a selected process.

The final test: Running with optimized I/O

After we adjusted the module search path for Python and removed the unused paths, the failed I/O numbers dropped significantly, and the runtime improved by 14 seconds. This performance improvement shaved costs by 2 percent when running the workload in isolation.

When running at scale, the savings are likely to be much higher, because the failed open() and stat() operations used shared storage. Running this application at scale would place a significant load on the NFS storage and quickly overwhelm the file system. For example, running 1,000 instances of this application would have generated more than half a billion failed I/O operations. That in turn would cause performance to degrade on the file system very quickly, leading to much slower runtimes as each failed I/O operation took longer.

Similar studies[1] have shown cost savings rise from 2 percent to 8 percent when scaling an application to just 16 nodes. Based on these numbers, it's unlikely that this customer's application could have run at the scales needed for EDA without the performance improvements highlighted by Breeze.

[1] I/O Profiling to Improve DL_POLY for Molecular Dynamics Simulation. Altair white paper. March 26, 2021.

Summary

Our semiconductor customer was satisfied that its power integrity and sign-off workload would run well in a hybrid cloud configuration. Callisto, the power-integrity workflow for this transistor design scenario, was copied bit by bit from the cluster on-premises to the Azure HPC environment. Afterwards, users didn't even realize that Callisto was running on Azure. They continued to use the same, familiar scripts and file system namespace that they'd used before on-premises.

We used Breeze like a detective to isolate the workflow segment most suited to Azure. Then, Breeze highlighted previously unknown performance problems, helping us make significant performance improvements for our customer's workload after it was deployed on Azure.

Azure HPC for silicon provides a globally available, reliable, scalable platform with compute, storage, and networking options optimized for EDA workloads. Microsoft values its partnerships in the silicon industry. We're working to improve the complex EDA software landscape and help semiconductor companies achieve better quality designs, innovative solutions, and faster product delivery.

Altair is a global technology company that provides software and cloud solutions in the areas of simulation, HPC, and AI. Altair enables organizations across broad industry segments to compete more effectively in a connected world while creating a more sustainable future.