Break through the limits of your GPU

Accelerated Computing with a Reconfigurable Dataflow Architecture

Trends Driving New Processing Architectures

With the rapid expansion of applications that can be characterized by dataflow processing, such as natural-language processing and recommendation engines, the performance and efficiency challenges of traditional, instruction set architectures have become apparent. To address this and enable the next generation of scientific and machine-learning applications, SambaNova Systems has developed the Reconfigurable Dataflow Architecture™, a unique vertically integrated platform that is optimized from algorithm to silicon. Three key long-term trends infused SambaNova's effort to develop this new accelerated computing architecture.

First, the sizable, generation-to-generation performance gains for multi-core processors have tapered off. As a result, developers can no longer depend on traditional performance improvements to power more complex and sophisticated applications. This holds true for both CPU fat-core and GPU thin-core architectures. A new approach is required to extract more useful work from current semiconductor technologies. Amplifying the gap between required and available computing is the explosion in the use of deep learning. According to a study by OpenAI, during the period between 2012 and 2020, the compute power used for notable artificial intelligence achievements has doubled every 3.4 months.

Second, is the need for learning systems that unify machine-learning training and inference. Today, it is common for GPUs to be used for training and CPUs to be used for inference based on their different characteristics. Many real-life systems demonstrate continual and sometimes unpredictable change, which means predictive accuracy of models declines without frequent updates. An architecture that efficiently supports both training and inference enables continuous learning and accuracy improvements while also simplifying the develop-train-deploy, machine-learning life cycle.

Finally, while the performance challenges are acute for machine learning, other workloads such as analytics, scientific applications and even SQL data processing all exhibit dataflow characteristics and will require acceleration. New approaches should be flexible enough to support broader workloads and facilitate the convergence of machine learning and HPC or machine learning and business applications.

Key Attributes for a Next-Generation Architecture

Through academic research, analysis of technology trends and knowledge developed in the design process, SambaNova identified the following key attributes to enable highly efficient dataflow processing.

- **Native dataflow** — Commonly occurring operators in machine-learning frameworks and DSLs can be described in terms of parallel patterns that capture parallelizable computation on both dense and sparse data collections along with corresponding memory access patterns. This enables exploitation and high utilization of the underlying platform while allowing a diverse set of models to be easily written in any framework of choice.
- **Support for terabyte-sized models** — A key trend in deep-learning model development uses increasingly large model sizes to gain higher accuracy and deliver more sophisticated functionality. For example, leveraging billions of data-points (referred to as parameters) enables more accurate Natural Language Generation. In the life sciences field, analyzing tissue samples requires the processing of large, high-resolution images to identify subtle features. Providing much larger on-chip and off-chip memory stores than those that are available on core-based architectures will accelerate deep-learning innovation.
- **Efficient processing of sparse data and graph-based networks** — Recommender systems, friend-of-friends problems, knowledge graphs, some life-science domains and more involve large sparse data structures that consist of mostly zero values. Moving around and processing large, mostly empty matrices is inefficient and degrades performance. A next-generation architecture must intelligently avoid unnecessary processing.
- **Flexible model mapping** — Currently, data and model parallel techniques are used to scale workloads across the infrastructure. However, the programming cost and complexity are often prohibiting factors for new deep-learning approaches. A new architecture should automatically enable scaling across infrastructure without this added development and orchestration complexity and avoid the need for model developers to become experts in system architecture and parallel computing.
- **Incorporate SQL and other pre-/post data processing** — As deep learning models grow and incorporate a wider variety of data types, the dependency on pre-processing and post-processing of data becomes dominant. Additionally, the time lag and cost of ETL operations impact real-time system goals. A new architecture should allow the unification of these processing tasks on a single platform.

A New Approach: SambaNova Reconfigurable Dataflow Architecture™

The SambaNova Reconfigurable Dataflow Architecture™ (RDA) is a computing architecture designed to enable the next generation of machine learning and high performance computing applications. The Reconfigurable Dataflow Architecture is a complete, full-stack solution that incorporates innovations at all layers including algorithms, compilers, system architecture and state-of-the-art silicon.

The RDA provides a flexible, dataflow execution model that pipelines operations, enables programmable data access patterns and minimizes excess data movement found in fixed, core-based, instruction set architectures. It does not have a fixed Instruction Set Architecture (ISA) like traditional architectures, but instead is programmed specifically for each model resulting in a highly optimized, application-specific accelerator.

The Reconfigurable Dataflow Architecture is composed of the following:
SambaNova Reconfigurable Dataflow Unit™ is a next-generation processor designed to provide native dataflow processing and programmable acceleration. It has a tiled architecture that comprises a network of reconfigurable functional units. The architecture enables a broad set of highly parallelizable patterns contained within dataflow graphs to be efficiently programmed as a combination of compute, memory and communication networks.

SambaFlow™ is a complete software stack designed to take input from standard machine-learning frameworks such as PyTorch and TensorFlow. SambaFlow automatically extracts, optimizes and maps dataflow graphs onto RDUs, allowing high performance to be obtained without the need for low-level kernel tuning. SambaFlow also provides an API for expert users and those who are interested in leveraging the RDA for workloads beyond machine learning.

SambaNova Systems DataScale™ is a complete, rack-level, data-center-ready accelerated computing system. Each DataScale system configuration consists of one or more DataScale nodes, integrated networking and management infrastructure in a standards-compliant data center rack, referred to as the SN10-8R.

Progress against the challenges outlined earlier would be limited with an approach the solely focuses on a new silicon design or algorithm breakthrough. Through an integrated, full-stack solution, SambaNova is able to innovate across layers to achieve a multiplying effect. Additionally, SambaNova DataScale leverages open standards and common form factors to ease adoption and streamline deployment.

Motivations for a Dataflow Architecture

Computing applications and their associated operations require both computation and communication. In traditional core-based architectures, the computation is programmed as required. However, the communications are managed by the hardware and limited primarily to cache and memory transfers. This lack of ability to manage how data flows from one intermediary calculation to the next can result in excessive data transfers and poor hardware utilization.

The SambaNova Reconfigurable Dataflow Architecture provides an alternative approach where the communications can be programmed and optimized to best suit how data should transit a series of computations. Figure 1 shows a few commonly occurring operations or parallel patterns. The orange circles show the computation needs that which are programmed on any architecture. More importantly, the arrows and boxes capture the dataflow required by each pattern's inputs and outputs. Even in these few examples, dataflow patterns vary widely as shown (e.g., one-to-one, many-to-one) demonstrating the opportunity that programmable dataflow can provide.

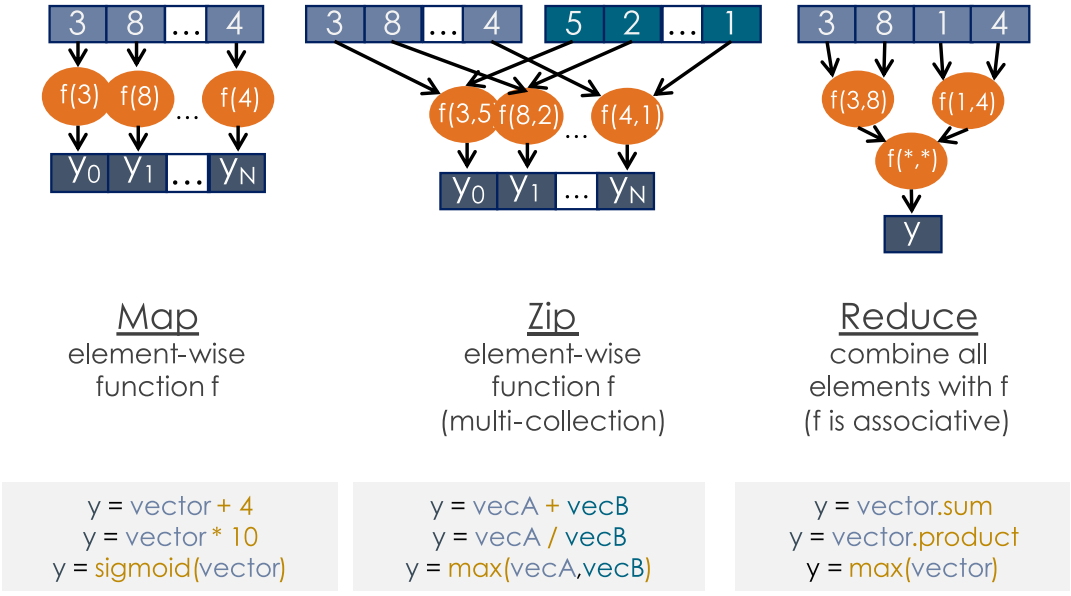

Figure 1 - Example Parallel Patterns: Map, Zip and Reduce

Dataflow programming of RDUs is provided by SambaFlow, which uses spatial programming. Spatial programming involves configuring the physical resources of the RDU so that data progresses efficiently in parallel across the fabric of the chip. Through fast reconfiguration, SambaFlow can also program the sequence of instructions (layers) running on the chip at a specific time. By incorporating both the sequence of instructions and the location of allocated resources, SambaFlow can determine the most efficient layout of a compute graph to create a pipelined accelerator specific to the desired workload. The impact is to achieve much higher throughput, higher hardware utilization and lower latency. In contrast, implementing a complex compute graph on core-based architecture requires executing a large number of sequential instructions, where there is no optimization of dataflow for a particular workload.

Overcoming Memory Bottlenecks with Dataflow

Traditional hardware architectures operate on a stream of low-level instructions that not only have poor energy efficiency, but also force a kernel-by-kernel programming model for dataflow-oriented workloads. For dataflow-oriented workloads such as machine learning and HPC, this kernel-by-kernel execution model can cause excess data and instruction movement that results in processor utilization, which is a small fraction of the theoretical peak FLOPs.

To look at this further, Figure 2 shows an extremely simple, logical compute graph for a convolution network that consists of just five kernels. Figure 3 shows the execution sequence of this convolution graph on a traditional core-based architecture. During execution, each kernel must be loaded onto the CPU or GPU, data and weights are read from memory, calculations are performed and the output results are written to memory. The process then repeats for each stage, multiplying the amount of data movement and consuming large amounts of memory bandwidth.

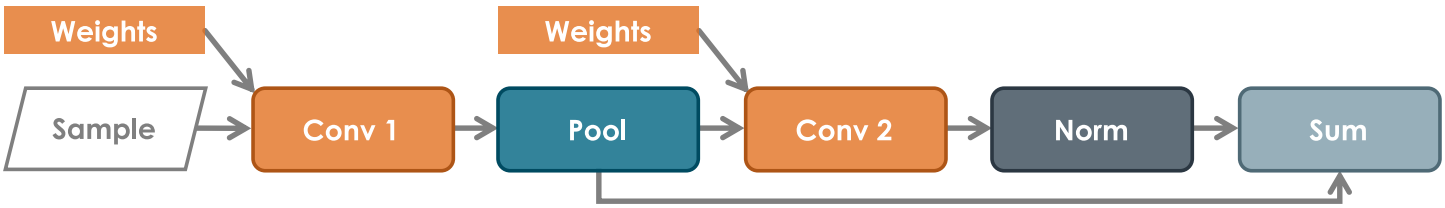

Figure 2 - Simple convolution graph

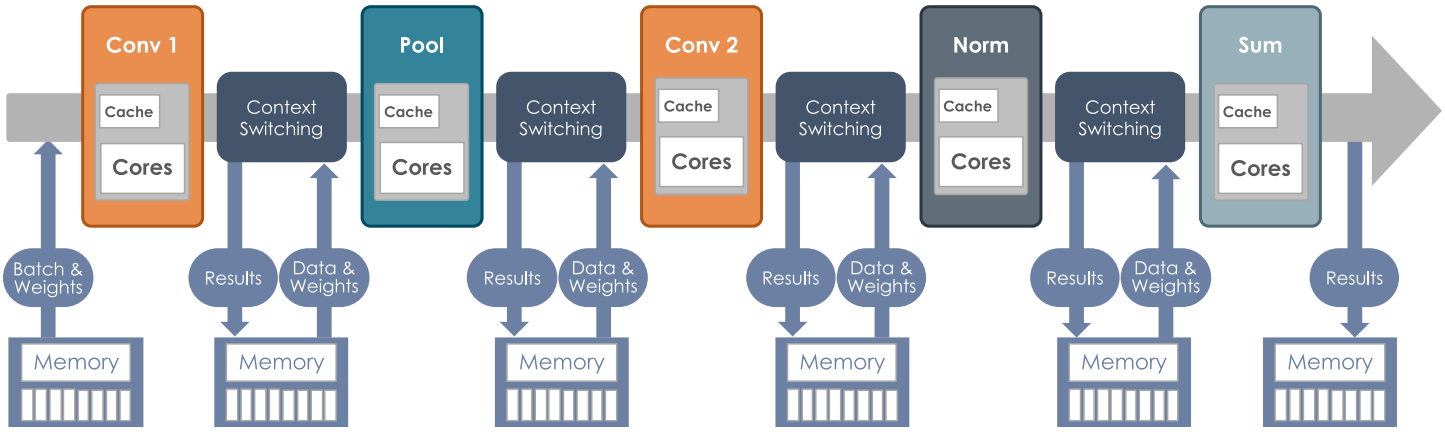

Figure 3 - Core-based kernel by kernel execution

For many workloads, adequate computing power may be available. However, excess data movement leads to poor hardware utilization and intractable training times. As a result, researchers and developers often need to limit their algorithm designs to those that they can afford to train.

In contrast, the SambaNova Reconfigurable Dataflow Architecture creates custom processing pipelines that allow data to flow through the complete computation graph. It uses a spatial programming model to optimize compute layout and minimize data movement to achieve high hardware utilization.

Figure 4 shows the execution of the same convolution graph on a Reconfigurable Dataflow Unit. The primary RDU elements, Pattern Compute Units (PCUs), Pattern Memory Units (PMUs) and the switch fabric that are described in more detail in the next section, provide the resources for the graph execution. These elements are programmed by SambaFlow to provide dataflow patterns such as many-to-one, one-to-many, broadcast, etc. as required to support each kernel's unique requirements. Spatial programming techniques are applied to ensure that the layout of the operations on the RDU minimizes data movement to achieve high efficiency.

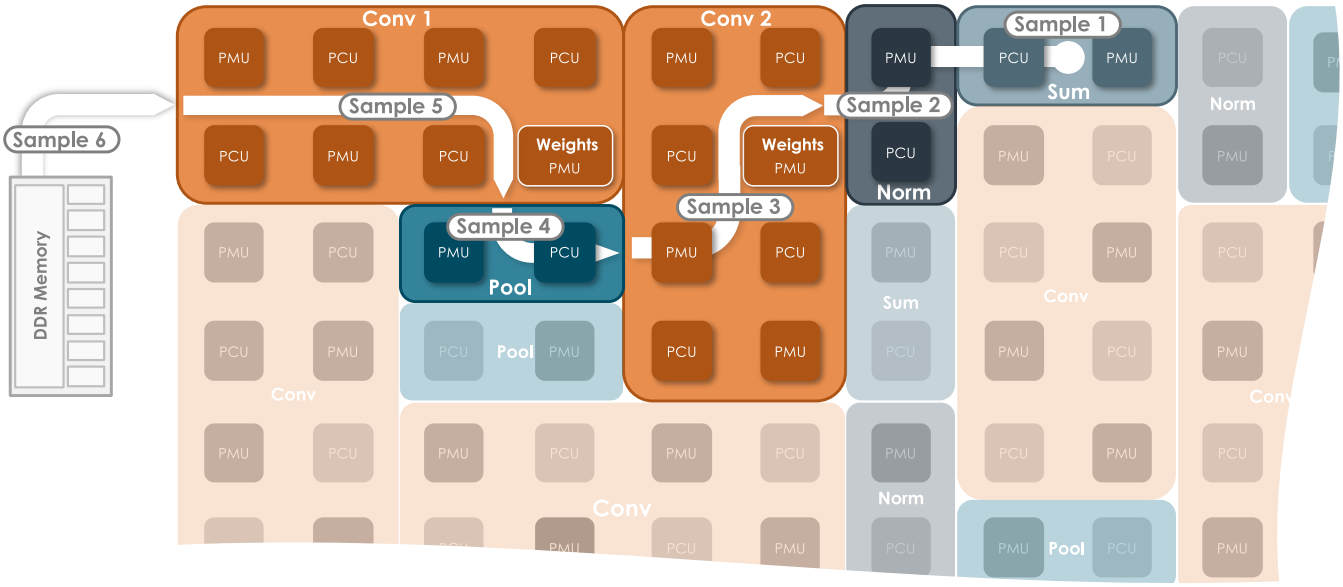

Figure 4 - RDU dataflow execution

When an application is launched, SambaFlow performs a one-time configuration to map the entire model onto RDUs. The entire system performs as a pipeline with different parts of the RDUs executing different layers of a model, working simultaneously with different data at each stage. Data is able to flow through each layer unobstructed and avoid the latency of context switching and memory access shown in Figure 3.

Flexibility and Reconfigurability with Dataflow

The SambaFlow optimizations described above and programmability of the RDU allow it to be optimized and configured for a variety of workloads across machine learning, scientific computing and other data-intensive applications. Rapid reconfiguration enables the architecture to be quickly repurposed for new needs or to adapt to the latest algorithm breakthroughs.

These are key advantages over fixed ASIC designs that can require years to develop and cannot be modified for algorithm changes or different workloads. At the other end of the spectrum, are FPGAs, which are highly reconfigurable. In contrast to the time-consuming, complex, low-level programming and long compilation times of FPGAs, RDUs can be reconfigured in microseconds.

This level of flexibility and reconfigurability gives programmers the ability to work in high-level DSLs while providing enhanced execution efficiency, simplified compilation and performance.

Advantages of the dataflow approach:

- Less data and code movement reduces memory bandwidth needs and enables the use of much larger, terabyte-sized attached memory for large model support.
- Simultaneous processing of an entire graph in a pipelined fashion enables high utilization across a broad range of batch sizes and eliminates the requirement to use large batch sizes to achieve acceptable efficiency.
- High on-chip memory capacity and localization, as well as high internal fabric bandwidth enable theability to run very large models at high performance.
- Pipeline processing on RDUs provides predictable, low-latency performance.
- Hierarchy of this architecture simplifies compiler mapping and significantly improves execution efficiency.

SambaNova Cardinal SN10™ Reconfigurable Dataflow Unit

SambaNova Systems Cardinal SN10™ Reconfigurable Dataflow Unit is the engine that efficiently executes dataflow graphs. The RDU consists of a tiled array of reconfigurable processing and memory units connected through a high-speed, three-dimensional on-chip switching fabric. When an application is started, SambaFlow configures the RDU elements to execute an optimized dataflow graph for that specific application. Figure 5 shows a small portion of an RDU with its components described below.

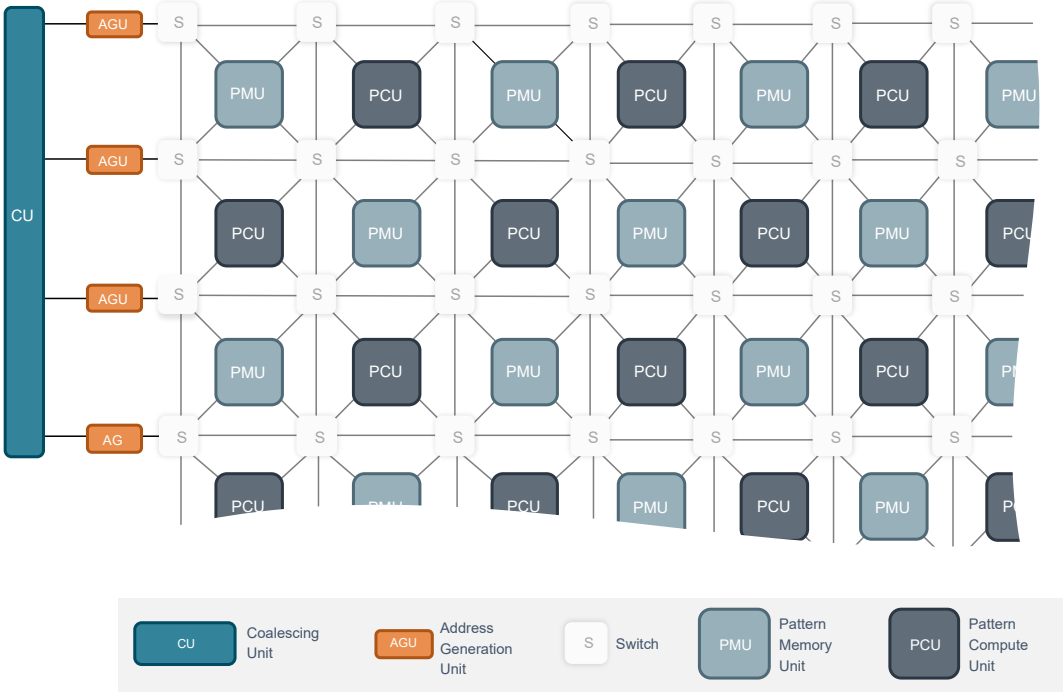

Figure 5 - Simplified RDU architecture and components

Pattern Compute Unit (PCU) — The PCU is designed to execute a single, innermost-parallel operation in an application. The PCU data-path is organized as a multi-stage, reconfigured SIMD pipeline. This design enables each PCU to achieve high-compute density and exploit both loop-level parallelism across lanes and pipeline parallelism across stages.

Pattern Memory Unit (PMU) — PMUs are highly specialized scratchpads that provide on-chip memory capacity and perform a number of specialized intelligent functions. The high PMU capacity and distribution throughout the PCUs minimizes data movement, reduces latency, increases bandwidth and avoids off-chip memory accesses.

Switching Fabric — The high-speed switching fabric that connects PCUs and PMUs is composed of three switching networks: scalar, vector and control. These switches form a three-dimensional network that runs in parallel to the rest of the units within an RDU. The networks differ in granularity of data being transferred; scalar networks operate at word-level granularity, vector networks at multiple word-level granularity and control at bit-level granularity.

Address Generator Units (AGU) and Coalescing Units (CU) — AGUs and CUs provide the interconnect between RDUs and the rest of the system, including off-chip DRAM, other RDUs and the host processor. RDU Connect™ provides a high-speed path between RDUs for efficient processing of problems that are larger than a single RDU. The AGUs and CUs working together with the PMUs enable RDA to efficiently process sparse and graph-based datasets.

Reconfigurability, exploitation of parallelism at multiple levels and the elimination of instruction processing overhead gives RDUs their significant performance advantages over traditional architectures.

SambaFlow™

SambaFlow™, the software component of the Reconfigurable Dataflow Architecture, is designed to be easy-to-use way to shield algorithm developers from low-level tuning needs that are common on other architectures. Users can maximize productivity by continuing to work in high-level frameworks like PyTorch and TensorFlow and not worry about architectural details of the RDU. It also provides an alternate User Graph interface for expert users and those who are interested in leveraging DataScale Systems for workloads beyond machine learning.

SambaFlow connects to machine-learning frameworks and analyzes models to build dataflow graphs. It then automatically decomposes the dataflow graphs with the knowledge of the resources required to execute the graph. This automated process results in a fully optimized, custom accelerator while avoiding low-level programming and time-consuming trial-and-error tuning. With complete a understanding of the software stack from the model down to the processing elements, SambaFlow's multi-stage tools have the advantage of being able to optimize at appropriate levels to run in the most efficient way.

SambaFlow also automates the scaling of workloads across multiple RDUs. By contrast, when working with large models on traditional architectures, a key challenge is using data and model parallel techniques to break the workload up and spread it across resources. Particularly for model parallel techniques, this requires developing external frameworks or using trial-and-error guesswork to split the model apart to achieve optimal results. Moving a model from a single processor to a large compute cluster often requires considerable extra development effort, orchestration and specialized expertise. Scaling out to a large cluster to achieve a high enough memory size can also result in a large sacrifice in utilization per device.

SambaFlow, however, provides a consistent programming model that spans from one RDU tile up to multi-system configurations. The ability of SambaFlow to automatically understand the underlying resources of the hardware and directly optimize a model provides the unique advantage of fully automating both multi-chip, data-parallel and model-parallel support. Developers allocate one or more RDUs to a model, then SambaFlow compiles to automatically provide the most efficient execution possible with the given set of resources. This enables developers to be more productive and reduces time to production.

SambaFlow also enables rapid reconfiguration of RDUs. For example, when model, data-source or batch-size changes are desired, RDUs can be reconfigured accordingly. Unlike FPGA programming, RDU reconfiguration is lightweight and can take 10-40 microseconds depending on model complexity.

SambaFlow has several components that optimize the application and manage system resources. These components are shown in Figure 6 and described below.

User Entry Points

- Write to popular ML frameworks
- Push-button automation path

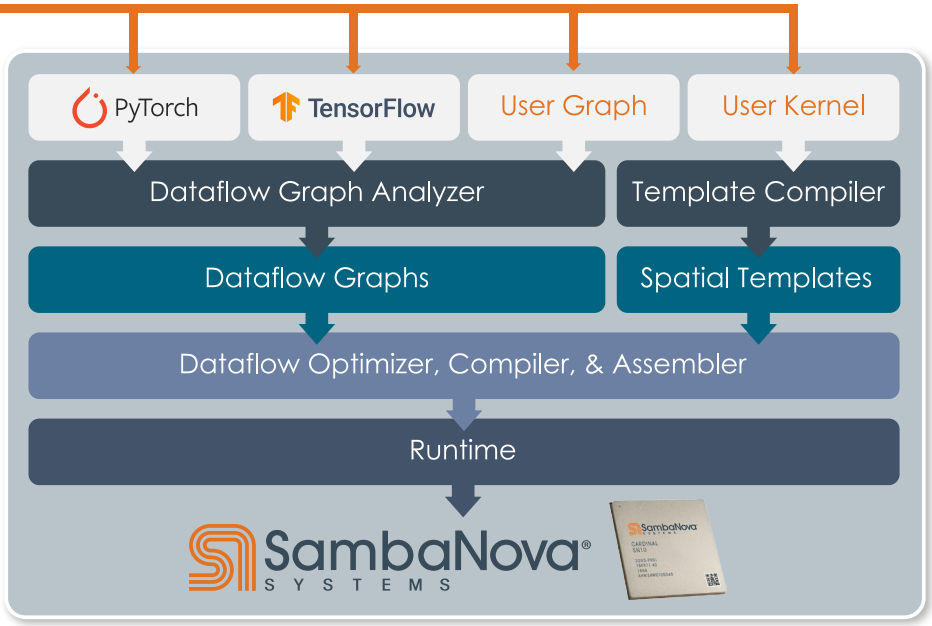

Figure 6 - SambaFlow components

User Entry Points – SambaFlow supports the common open-source, machine-learning frameworks, PyTorch and TensorFlow. Serialized graphs from other frameworks and tools are also imported here.

Dataflow Graph Analyzer and Dataflow Graphs — Accepts models from the frameworks then analyzes the model to extract the dataflow graph. For each operator, the computation and communication requirements are determined, so the appropriate RDU resources can be allocated later. The analyzer determines the most efficient mappings of the operators and communication patterns to the RDU utilizing the spatial programming model. With knowledge of both the model architecture and the RDU architecture, the analyzer can also perform high-level, domain-specific optimizations like node fusion. The output of the Dataflow Graph Analyzer is an annotated Dataflow Graph that serves as the first intermediate representation (IR) passed to the Dataflow Compiler.

Template Compiler and Spatial Templates — For cases where operators are required but not available in the existing frameworks, new operators can be described via a high-level, tensor index notation API. The Template Compiler will then analyze the operator and generate an optimized dataflow implementation for the RDU, called a Spatial Template. The generated template includes bindings that enable the new operator to be used directly from application code in the same way as built-in framework operators.

Dataflow Compiler, Optimizer and Assembler — This layer receives annotated Dataflow Graphs and performs high-level transformations like meta-pipelining, multi-section support and parallelization. It also understands the RDU hardware attributes and performs low-level transforms, primarily placing and routing by mapping the graph onto the physical RDU hardware and then outputting an executable file. As before, a spatial programming approach is used to determine the most efficient location of RDU resources.

Advantages of SambaFlow:

- **Support for popular open-source ML frameworks** such as PyTorch and Tensorflow.
- **Tight integration between algorithms, SambaFlow and RDUs** results in a custom, highly parallelized accelerator that is uniquely optimized for each specific model.
- **Push-button model compilation and optimizations** allow high performance to be obtained out of the box without the need for hand tuning. This allows rapid experimentation with high-performance models without deep-performance and hardware-tuning expertise.
- **Automated data and model parallel mapping** simplifies scaling from test bed to large-scale production by using the same programming model as on a single device, without requiring the kind of careful and tedious internode cluster programming needed for memory-constrained devices.
- **Spatial programming model** automatically assigns resources in the most efficient manner to provide a complete processing pipeline that enables high RDU utilization by minimizing data movement and off-chip accesses.
- **Secure multi-tenancy and concurrent multi-graph execution** provides seamless scale-up and scale-out flexibility for best utilization of resources. Application isolation support ensures not just multiple applications but also multiple users, and thus support for resource management tools such as Slurm.
- **High-performance data transfer** with protocol support for RoCE, RDMA, Ethernet and Infiniband protocols ensures that models requiring data transfer between devices can run at full speed.
- **Virtualization and container support** means you can securely use and deploy Docker, Kubernetes or VM environments.

Large Model Use Cases on RDA

Across a variety of applications, there is a trend towards models that require large memory capacities although the drivers vary by usage. For example, in NLP, a very large number of parameters enables the summarization of complex text passages or generation of improved text suggestions. With recommender systems, larger memory sizes support rich user and catalog embeddings to achieve higher recommendation accuracy and conversion rates. In computer vision, larger memory is required for the high-resolution images and associated activations for uses like pathology, medical scans and astronomy.

To support large models, a common approach is to partition a model up into small parts each of which fitting into GPU memory. The latest large NLP models are often trained on configurations built with thousands of GPUs. Sheer size, cost and complexity serve to limit innovation to large, well-funded organizations.

By contrast, massive models that previously required 1,000+ GPUs to run are able to run on a single SambaNova Datascale

System. This is made possible by the dataflow processing model and large on-chip capacity that reduces off-chip communication and pressure on memory bandwidth and has allowed the use of terabyte-sized attached memory. Running models across multiple RDUs in a system is also simplified through automated data and model parallel scaling.

SambaNova publishes articles regularly about large models, scaling, performance and other topics. To learn more, please see https://sambanova.ai/articles/.

Multi-Tenant and Concurrent Applications on RDA

While DataScale systems can be used for large-scale applications as described earlier, they can also support multiple concurrent applications and provide multi-tenant isolation as shown in Figure 7. Teams or applications may only require a portion of a system, and organizations can use the multi-tenant functionality to provide machine-learning, private-cloud resources that serve multiple departments or customers. It is also possible to dedicate some portion of the RDU to training updated models while other portions execute previously trained models for inferencing and results generation.

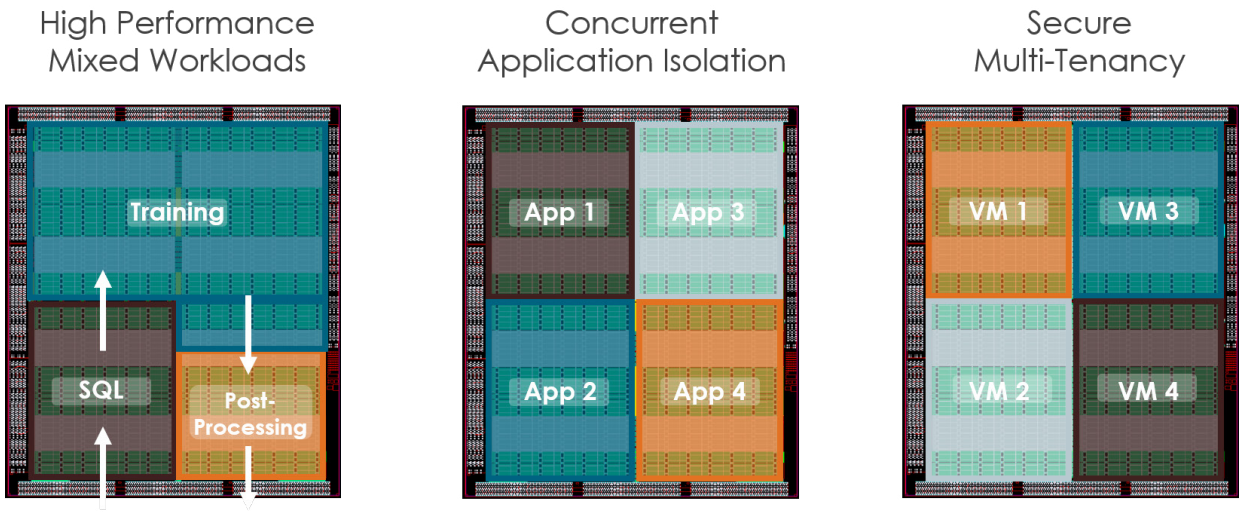

Figure 7 - Supporting multiple users or workloads simultaneously

Model Life Cycle Management on RDA

Algorithm development is, of course, only a small part of successful production operations as shown in Figure 8. Capabilities of the RDA described earlier can simplify and facilitate additional parts of the machine-learning life cycle.

For example, RDUs enable the integration of pre- and post-processing to avoid additional ETL overhead. Reconfigurability facilitates rapid iteration on model development by experimentation and adjustments to models in the development and testing phase. Support in RDUs for both high throughput, large batch-size training as well as small batch-size inference means that models can combine training and inference in a continuous learning or incremental training mode. This allows for new and interesting types of models to be developed that are not possible in conventional life cycle flows.

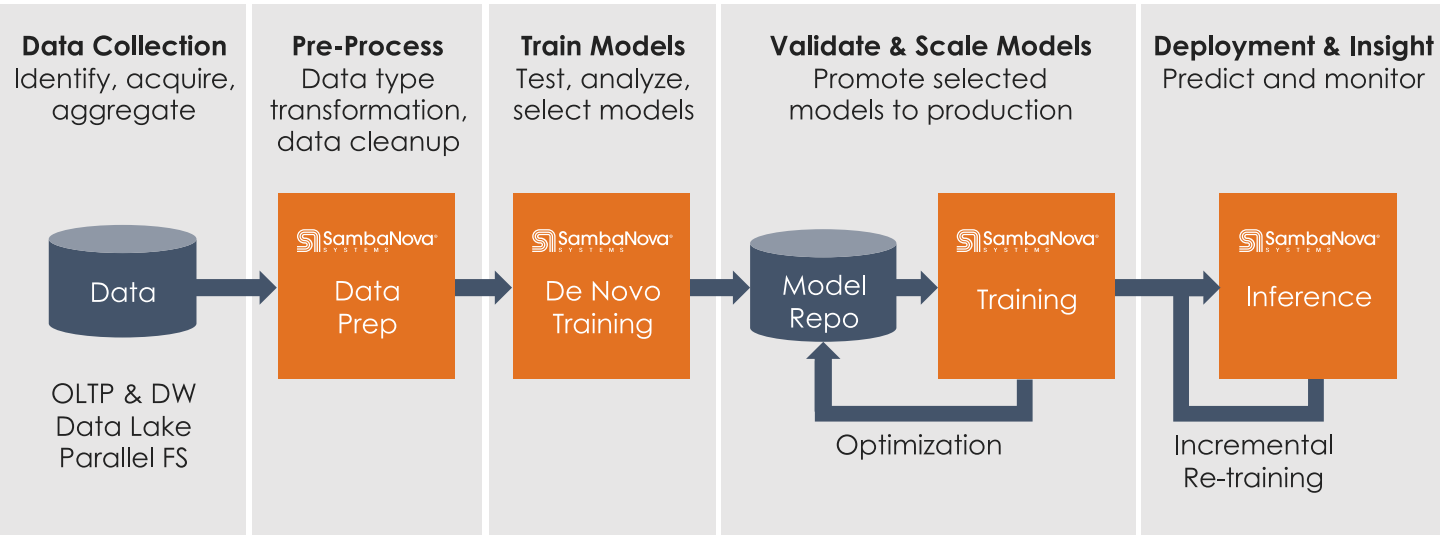

Figure 8 - Optimization across the complete life cycle

Summary

SambaNova DataScale systems and SambaFlow provide a unique vertically integrated platform that is optimized from the algorithm, through the compiler, and down to the silicon to achieve breakthrough performance and flexibility for innovators creating the next generation of machine-learning and deep-learning applications. Contact SambaNova to learn more about accelerating your machine-learning and HPC applications.

For more information visit sambanova.ai
or call +1 650 263-1153 to speak to a
SambaNova representative

sambanova.ai