HammerBlade Architectural Overview

HammerBlade is a data-processing architecture that can scale from super-computer systems to mobile devices. Each HammerBlade **Node** is a single System-on-Chip and multiple HammerBlade Nodes are interconnected with high-bandwidth (100G+) links. A HammerBlade Node is architected from an an array of tiles connected by a 2-D mesh network, called the **manycore accelerator network** (sometimes referred to as on-chip network, NOC, OCN, “the network” etc.), and is attached to a reconfigurable high-bandwidth memory system and the I/O system. Tiles within a HammerBlade Node are processing elements, memory, or IO interfaces.

Processing tiles fit into one of three categories: **throughput-optimized** RISC-V tile, a **linux-capable** RISC-V core, or a specialized **accelerator**. The throughput-optimized cores in our system are called *Vanilla-5 (V5)* cores. Each Vanilla-5 core supports floating-point operations, contains a scratchpad for local data storage, and an instruction cache. The Linux-capable cores are open-source 64-bit BlackParrot RISC-V cores from the University of Washington, which are being co-developed in the DARPA POSH program. The accelerator cores customize the HammerBlade architecture and to improve energy/performance for high-value applications.

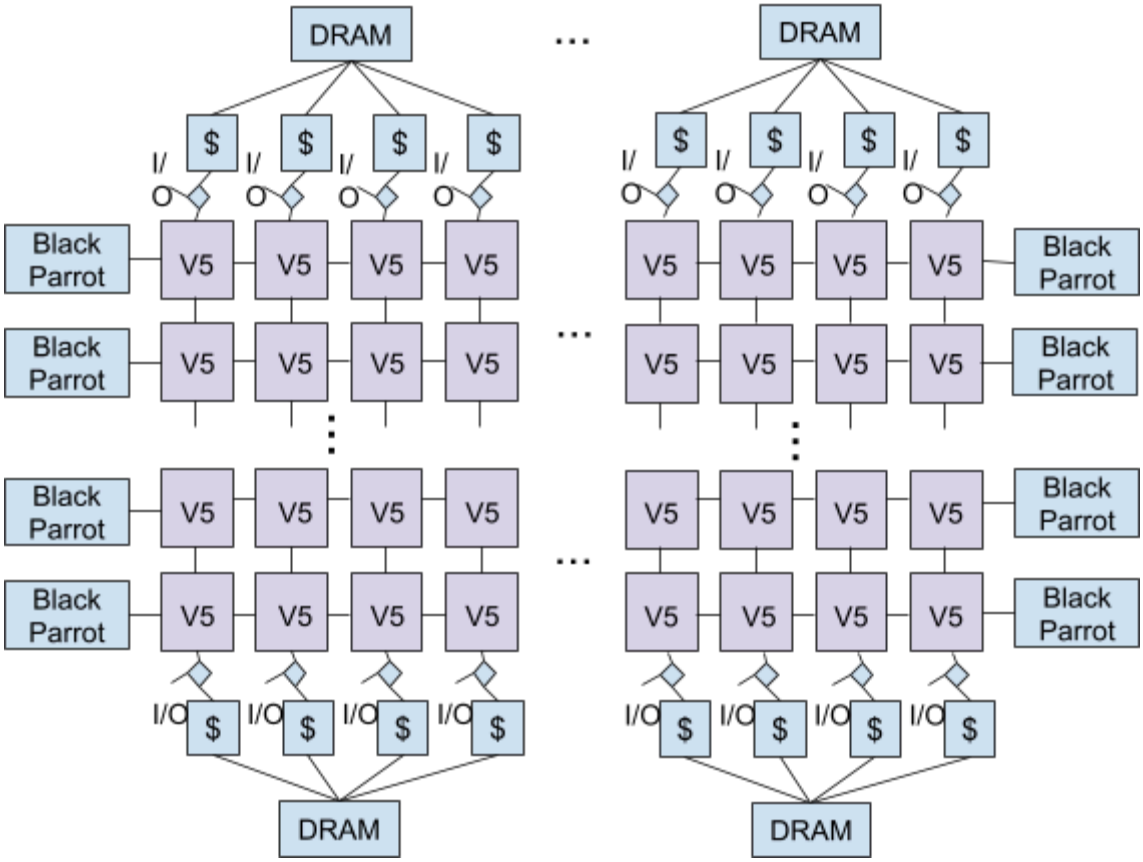
HammerBlade **Accelerators** are diverse, flexible and performant. In collaboration with Cornell University, we are developing a CGRA fabric for accelerating large sequences of branch-free code with instruction-level parallelism and a sparse-tensor architecture for accelerating sparse linear algebra operations. Together, with dense-tensor accelerators we have developed at UW these will provide efficient computation for high-value workloads in the SDH program. These accelerators will have seamless software integration with other tiles on the chip.

Cache ($) tiles provide the interface to external memory. The cache is reconfigurable and introspective. These tiles are typically located on the top and bottom edge of a tile array and are called column-caches. These column cache tiles are connected to memory controllers that interface to multiple parallel memory channels of a DRAM-based memory system -- high bandwidth memory (HBM), DDR4, GDDR5, etc. Applications will be able to measure performance and reconfigure the memory system parameters depending on the demands of the phase of an application.

A collection of tiles is called a **Pod**, and a HammerBlade Node contains multiple Pods. Pods impose a hierarchy in communication: Intra-Pod communication will have lower latency and higher-bandwidth than inter-Pod communication. Likewise, inter-Pod (intra-Node) communication will have lower latency and higher bandwidth than inter-Node communication. The Pod dimensions, distribution of tile types, cache sizes, number of memory DRAM interface channels, and mapping from channels to caches is flexible and an active area of research

The HammerBlade address space is **non-uniform and globally addressable**; all tiles can read and write memory of other tiles and external memory. Tiles can read and write external memory by sending requests to the cache interfaces. Pairs of tiles can communicate by sending read and write requests to globally-mapped memory ranges of other tiles. These features allow tiles to form efficient software abstractions for data communication patterns.

The figure below shows the high-level architecture of a HammerBlade Pod (without accelerators).



HammerBlade Software Overview

The HammerBlade software paradigm is **Single-Program Multiple Data** (SPMD). Abstractly, the user writes a number of application kernels, and allocates tiles to execute those kernels using the HammerBlade CUDA-Lite Runtime. The CUDA-Lite runtime allows users to read and write data in device DRAM, enqueue applications for launch, launch applications, and copy results back to the host.

The work-unit of HammerBlade CUDA-Lite is the **Tile Group**. A Tile Group is a two-dimensional collection of tiles that execute a single program. All tiles in a tile group operate concurrently. Multiple Tile Groups may execute simultaneously depending on available resources. Tile Groups of different dimensions and unrelated programs may execute concurrently. Tile Groups do not have an execution order unless dependencies are explicitly specified. Intra-Tile-Group communication is supported by hardware, while Inter-Tile-Group communication is not. Each tile within a Tile Group is assigned a Tile Group ID that corresponds to its location within the tile group. All tiles within a tile group run the same program, and use their Tile Group ID to determine what subset of data to compute.

Iterative invocations of Tile Groups are launched using **Grids**. To achieve maximum performance over an entire application a Tile Group may be limited to computing to a subset of the input or output data. Grids allow users to specify iterative invocations of a Tile Group to compute a complete result. Tile Groups run by a grid are subject to the same ordering and communication constraints as all Tile Groups.

Related Reading

The Celerity Open-Source 511-Core RISC-V Tiered Accelerator Fabric.

This document describes the pre-HammerBlade architecture; many features have been added to the manycore, including i-cache, non-blocking loads, victim caches on each manycore column, and direct access to HBM2 channels. Additionally, the Rocket cores and neural networks have been removed.

The BaseJump Manycore Accelerator Network

This document describes how to interface to the manycore network. The most up-to-date version is here.