Heterogeneous Computing: Integration & Disaggregation

- hadoop-style workloads  
  ... scale-out via network

- main metrics
  - cost (capital, energy)
  - compute density
  - scalability

  → homogeneous nodes
  (CPU / FPGA / NVMe plus compute)

  → datacenter disaggregation

- complex HPC-like workloads  
  ... scale-up via high-speed buses

- main metrics
  - memory / accelerator / inter-node BW
  - optimal mix of heterogeneous resources
    (CPU / GPU / FPGA / HBM / DRAM / NVMe)
  - compute density, scalability

  → heterogeneous nodes

  → data centric design

3/7/2017
Heterogeneous Nodes: POWER8 Accelerator Interfaces

- GPU (via NVLINK)
- POWER8+ Processor
- IBM & Partner Devices
- Memory Interface Control
- Server Class Memory

Connectors:
- 2 x 40 GB/s NVLINK
- 8 x 28.8 GB/s DMI
- 2 x 16 GB/s CAPI
POWER8 Memory System

- 8 high speed channels, 230 GB/s sustained memory BW
- 32 total DDR ports yielding 410 GB/s peak at the DRAM
- 1 TB memory capacity per fully configured processor socket
Near-Memory Acceleration on ConTutto

POWER8™ processor
Near-Memory Acceleration on ConTutto

Memory DIMMs  -->  Centaur  -->  ConTutto FPGA  -->  New Memory Technologies

Memory DIMMs  -->  Centaur  -->  Near-Memory Acceleration  -->  Memory DIMMs

Memory DIMMs  -->  Centaur  -->  Memory DIMMs

Memory DIMMs  -->  Centaur  -->  Memory DIMMs

Memory DIMMs  -->  Centaur  -->  Memory DIMMs
Heterogeneous Nodes: POWER8 Accelerator Interfaces

- **GPU** connected to the **POWER8+ Processor** via **NVLink** (2 x 40 GB/s)
- **POWER8+ Processor** connected to **Memory Interface Control** (8 x 28.8 GB/s via DMI)
- **IBM & Partner Devices** connected to **Memory Interface Control** (2 x 16 GB/s via CAPI)
- **Memory Interface Control** connected to **Server Class Memory**
Near-memory Processing

- Big-data analytics, neural networks, cognitive computing, graph algorithms, ... benefit from low latency, small access granularity, and large memories.
- Memory performance and power depend on a complex interaction between workload and memory system:
  - Locality of reference, access patterns/strides, ...
  - Cache size, associativity, replacement policy, ...
  - Bank interleaving, refresh, row buffer hits,...
- Current systems use “bare metal” programming to adapt workload to memory system.
- Memory system should be programmable / adaptive.
- Must integrate programmable compute capabilities to achieve substantial performance & power gains for a wide range of workloads.
Integrating Near-data Processing in a (POWER) Server

- enabling near-data processing capabilities, while being minimally-invasive, in an existing CPU architecture
- ability to implement wide range of near-data processing functionality from optimized fixed-function hardware to a multiprocessor SOC
- dereferencing all virtual pointers of the host process on the NDP, coherent with the CPUs view of the memory

Diagram:
- NDP
- CPU
- Memory
- NDP Manager
- System bus
- High-speed link
- CPU core
- NDP Access Point
results obtained on a system-simulator capable of both functional verification and performance estimations was developed

the Graph500 benchmark benefits from a low latency and small access granularity: NDP cores four times slower than the CPU cores outperform them for large problems

the NDPS show much better bandwidth utilization due to the small access granularity
Heterogeneous Computing: Integration & Disaggregation

- hadoop-style workloads ... scale-out via network
  - main metrics
    - cost (capital, energy)
    - compute density
    - scalability
  → homogeneous nodes (CPU / FPGA / NVMe plus compute)
  → datacenter disaggregation

- complex HPC-like workloads ... scale-up via high-speed buses
  - main metrics
    - memory / accelerator / inter-node BW
    - optimal mix of heterogeneous resources (CPU / GPU / FPGA / HBM / DRAM / NVMe)
    - compute density, scalability
  → heterogeneous nodes
  → data centric design
ZRL “Dome” μServer of Hyperscale DCs

- Cloud economics
  - density (>1000 nodes / rack)
  - integrated NICs
  - switch card (backplane, no cables)
  - medium to low-cost compute chips
- Passive liquid cooling
  - ultimate density (cooling >70W / node)
  - energy re-use
- Built to integrate heterogeneous resources
  - CPUs
  - Accelerators
Enabling factor for this is the special cooling plate. Features of this cooling plate are:
- Material with high thermal conductivity
  - For low power nodes: Copper
  - For higher power nodes: Copper with granulate of material with even higher thermal conductivity, e.g. Carbon
- Built-in heat pipes for even better thermal heat transfer from PCB to liquid cooling bulk heads
- Layered structure, with electrical insulation between layers, which enables
  - Power supply connectivity from the liquid cooling bulk heads to the PCB
  - Thereby avoiding large currents in base board (for reduced layer count, thickness and cost)

The standardized connector further enables HETEROGENEOUS SYSTEMS!
- Use of different part numbers (= types of compute nodes) in the base board slots
  - In particular GP/GPU enabled nodes, FPGA nodes, storage nodes (and any combination thereof)
CloudFPGA: Network-attached FPGAs in Hyperscale DCs

- **Disaggregation of compute resources**
  - FPGAs can be deployed independent of:
    - the # CPUs (respectively servers)
    - the server form factor (which keep on shrinking)
  - FPGAs can be provisioned / rented similar to other cloud compute, storage and network resources

- **Scalability**
  - Users can build SDN fabrics of FPGAs in the cloud
  - FPGAs are promoted to the rank of peer processor (end of slavery)
  - HW-based FPGA-to-FPGA communication provides low latency and high-Tput (RDMA NICs)
Reference Prototype:
Half-wide Baseboard w/ Switch & Compute Modules

u-switch module

Compute node

Next gen Cloud & IoT Edge Compute sled
Reference Prototype: FPGA Compute Node

- KU060 FPGA w/ 16GB memory, 10GbE, PCIe extension, board management controller
- The iNIC enables the FPGA to hook itself to the network and to communicate with other DC resources, such as servers, disks, I/O and other FPGA appliances
Hyperscale FPGA + NVMe

- XCKU060 FPGA
- PCIe3 x4
- 10GE
- DDR4 8GB
- 512GB NVMe
- x8 16Gb/s
- x2 NVMe

3/7/2017
CloudFPGA: Network-attached FPGAs in Hyperscale DCs
Distributed Shared Storage - DSS

... high speed, low latency access to a unified shared storage pool at single byte access granularity

- **Local NVMe attachment**
  - Generic device driver
  - Also works for any block dev
  - OFA RDMA interface

- **Global sharing**
  - Global access space
  - IB/RDMA integration
  - Flexibly configurable
  - User level dssd process
  - dssdsh command shell
Dense Storage: Software Components

- **3 kernel modules**
  - dsa.ko, sal.ko, sal_blkdev.ko
  - DSS GSL part of SAL
- **1 user library**
  - libdsa
- **1 user level demon**
  - dssd
Flexible DSS Configuration

- mix of local and shared resources
- multiple shared DM partitions possible
Basic Concepts DSS/GSL (1)

- SAL/DSS represents Storage-Class-Memory (SCM) devices (e.g. NVMe devices) as partitions, each providing byte-addressable linear memory regions.

- A partition consists of an array of Virtual Containers (VC).

- VCs reside in Physical Containers (PC) that represent physical memory or storage resources.
Basic Concepts DSS/GSL (2)

- A NVMe device is partitioned into one or more PCs.
  - A PC represents a physical resource and is tied to it

- Each PC is partitioned further into one or more VCs.
  - A VC is movable and can be on different physical resources at different times.
  - Purpose: Resiliency, Load balancing
DSS on 16 Nodes / 16 Clients

- random read
- random write
- sequential read
- sequential write
- mixed 50% read/write (IOPs)
But be willing to take incremental steps when you can!