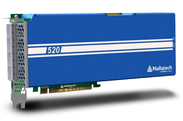
Supercomputer at CCS: Cygnus

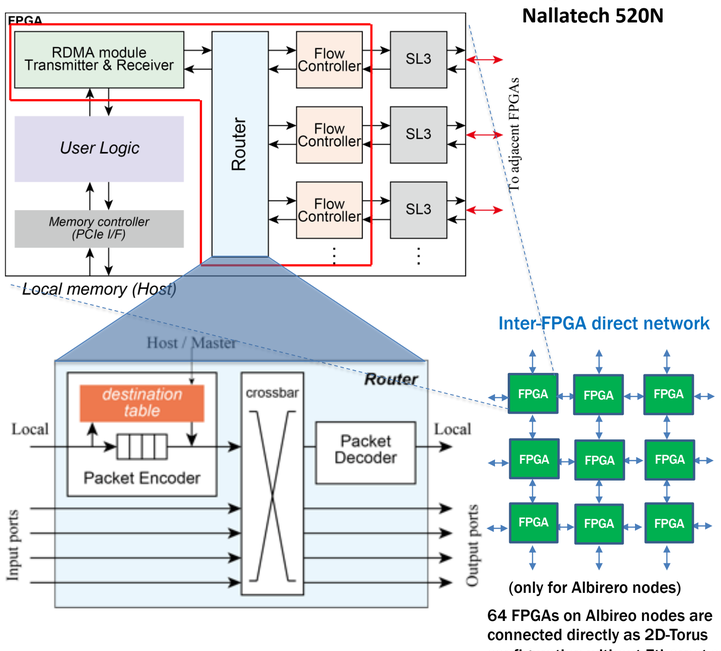
Multi-Hybrid Accelerated Computing Platform

- Combining goodness of different type of accelerators: GPU + FPGA
  - GPU is still an essential accelerator for simple and large degree of parallelism to provide ~10 TFLOPS peak performance
  - FPGA is a new type of accelerator for application-specific hardware with programmability and speeded up based on pipelining of calculation
  - FPGA is good for external communication between them with advanced high speed interconnection up to 100Gbps x4 chan
- Our new supercomputer "Cygnus"
  - Operation started in May 2019
  - 2x Intel Xeon CPUs, 4x NVIDIA V100 GPUs, 2x Intel Stratix10 FPGAs
  - Deneb: 48 CPU+GPU nodes
  - Albireo: 32 CPU+GPU+FPGA nodes with 2D-torus dedicated network for FPGAs (100Gbpsx4)
- FPGA design plan
  - Router
  - For the dedicated network, this impl. is mandatory.
  - Forwarding packets to destinations
  - User Logic
  - OpenCL kernel runs here.
  - Inter-FPGA comm. can be controlled from OpenCL kernel.
  - SL3
  - SerialLite III : Intel FPGA IP
  - Including transceiver modules for Inter-FPGA data transfer.
  - Users don't need to care

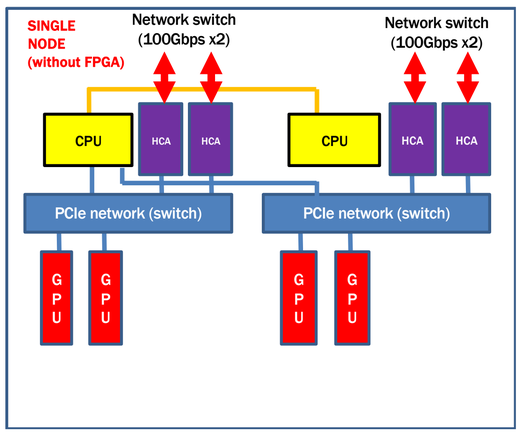
Target GPU: NVIDIA Tesla V100



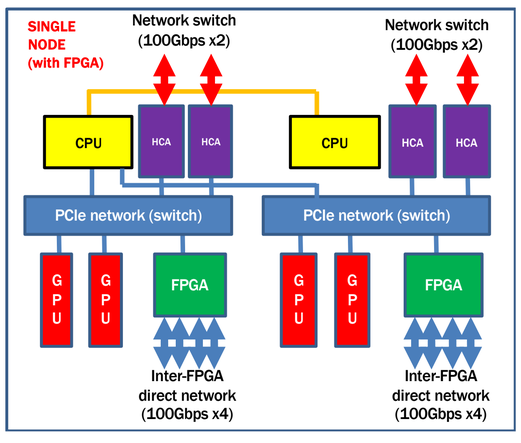
Target FPGA: Nallatech 520N



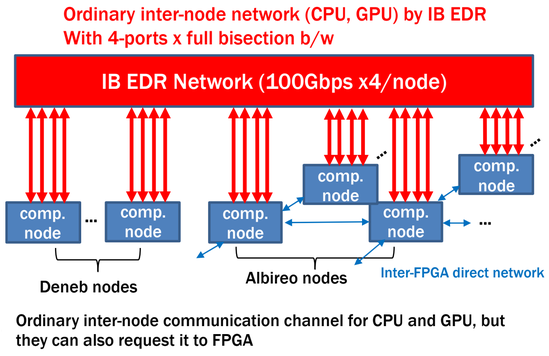
64 FPGAs on Albireo nodes are connected directly as 2D-Torus configuration without Ethernet sw.



Deneb node (x48)



Albireo node (x32)



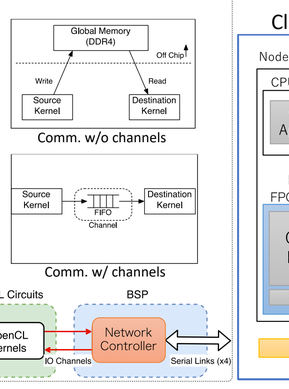
Ordinary inter-node communication channel for CPU and GPU, but they can also request it to FPGA

Specification of Cygnus

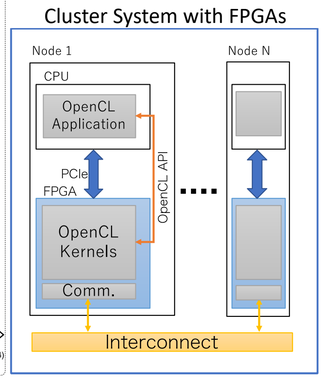
Item	Specification
Peak performance	2.4 PFLOPS DP (GPU: 2.2 PFLOPS, CPU: 0.2 PFLOPS, FPGA: 0.6 PFLOPS SP) ⇨ enhanced by mixed precision and variable precision on FPGA
# of nodes	80 (32 Albireo (GPU+FPGA) nodes, 48 Deneb (GPU-only) nodes)
Memory	192 GiB DDR4-2666/node = 256GB/s, 32GiB x 4 for GPU/node = 3.6TB/s
CPU / node	Intel Xeon Gold (SKL) x2 sockets
GPU / node	NVIDIA V100 x4 (PCIe)
FPGA / node	Intel Stratix10 x2 (each with 100Gbps x4 links/FPGA and x8 links/node)
Global File System	Lustre, RAID6, 2.5 PB
Interconnection Network	Mellanox InfiniBand HDR100 x4 (two cables of HDR200 / node) 4 TB/s aggregated bandwidthj
Programming Language	CPU: C, C++, Fortran, OpenMP, GPU: OpenACC, CUDA FPGA: OpenCL, Verilog HDL
System Vendor	NEC

OpenCL-ready High Speed FPGA Networking [1]

- FPGA-based parallel comp. with OpenCL
  - Needs a communication system being suitable to OpenCL and Intel FPGAs
  - Using of Intel FPGA SDK for OpenCL
    - Channel Extension: Transferring data between kernels directly (low latency and high bandwidth)
    - We can use multiple kernel design to exploit space parallelism in an FPGA
    - I/O Channel
      - connects OpenCL with peripherals
      - We used this feature



Cluster System with FPGAs



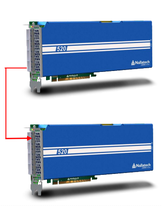
Our proposed method

- Communication Integrated Reconfigurable CompUting System (CIRCUS)
  - ➢CIRCUS enables OpenCL code communicate with other FPGAs on different nodes
  - ➢Extending Intel's channel mechanism to external communications
  - ➢Pipeline manner: sending/receiving data from/to compute pipeline directly

```
sender(__global float* restrict x, int n) {
  for (int i = 0; i < n; i++) {
    float v = x[i];
    write_channel_intel(simple_out, v);
  }
}
```
sender code on FPGA1

```
receiver(__global float* restrict x, int n) {
  for (int i = 0; i < n; i++) {
    float v = read_channel_intel(simple_in);
    x[i] = v;
  }
}
```
receiver code on FPGA2

CIRCUS Backends

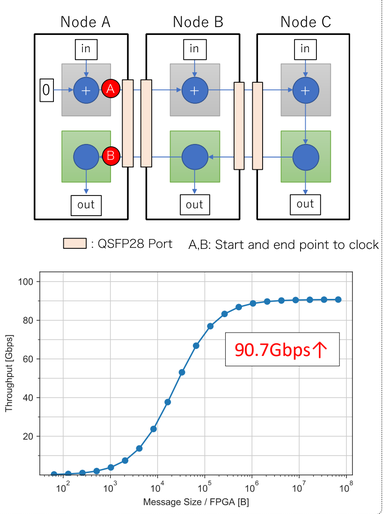


Pipelined communication experiment

```
uint16 val = (uint16)(0);
if (in_port == 1) {
    val = read_channel_intel(fwd_x_neg_in);
} else if (in_port == 2) {
    val = read_channel_intel(fwd_x_pos_in);
} else if (in_port == 3) {
    val = read_channel_intel(fwd_y_neg_in);
} else if (in_port == 4) {
    val = read_channel_intel(fwd_y_pos_in);
}
```
Recv.

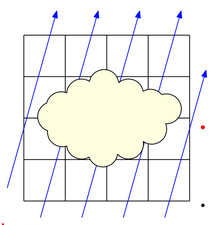
```
val += (uint16)(
    v + 0, v + 1, v + 2, v + 3,
    v + 4, v + 5, v + 6, v + 7,
    v + 8, v + 9, v + 10, v + 11,
    v + 12, v + 13, v + 14, v + 15
);
```
Comp.

```
ulong t_tmp = 0;
if (out_port == 1) {
    write_channel_intel(fwd_x_neg_out, clocktime(
        val, &t_tmp));
} else if (out_port == 2) {
    write_channel_intel(fwd_x_pos_out, clocktime(
        val, &t_tmp));
} else if (out_port == 3) {
    write_channel_intel(fwd_y_neg_out, clocktime(
        val, &t_tmp));
} else if (out_port == 4) {
    write_channel_intel(fwd_y_pos_out, clocktime(
        val, &t_tmp));
} else if (out_port == 5) {
    write_channel_intel(internal, clocktime(val, &
        t_tmp));
}
```
Send

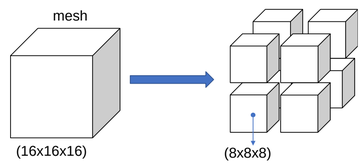


Authentic Radiation Transfer [2]

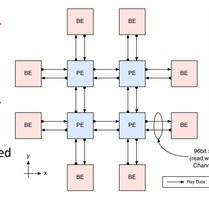
- Accelerated Radiative transfer on grids Oct-Tree (ARGOT) has been developer in Center for Computational Sciences, University of Tsukuba
  - ART is one of algorithms used in ARGOT and dominant part (90% or more of computation time) of ARGOT program
- ART is ray tracing based algorithm
  - problem space is divided into meshes and reactions are computed on each mesh
  - Memory access pattern depends on ray direction
  - Not suitable for SIMD architecture

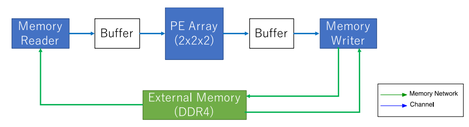


- Problem space is divided into small blocks
  - e.g. (16, 16, 16) → 8 × (8, 8, 8)
  - PE is assigned to each of small blocks

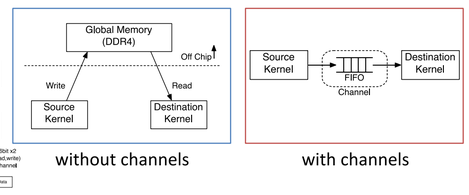


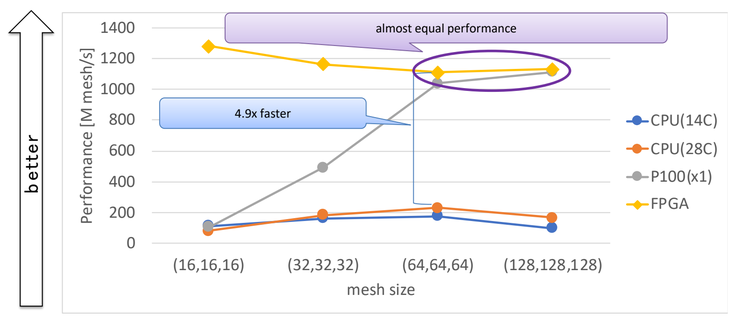
- PEs are connected by channels each other
  - PE: Processing Element
  - BE: Boundary Element
- Kernel of PEs and BEs are started automatically by autorun attribute
  - Lower control overhead and resource usage because of decreasing number of host controlled kernels





- Our implementation uses channel based approach
  - One of extensions to OpenCL for FPGAs by Intel
- It enables inter kernel communication much faster
  - No external memory (DDR) access is required
  - Lower resource utilization than DDR access





Size	CPU(14C)	CPU(28C)	P100	FPGA
(16,16,16)	112.4	77.2	105.3	1282.8
(32,32,32)	158.9	183.4	490.4	1165.2
(64,64,64)	175.0	227.2	1041.4	1111.0
(128,128,128)	95.4	165.0	1116.1	1133.5

Reference
[1] Norihisa Fujita, Ryohei Kobayashi, Yoshiki Yamaguchi, and Taisuke Boku, Parallel Processing on FPGA Combining Computation and Communication in OpenCL Programming, 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp.479-488, May 2019
[2] Norihisa Fujita, Ryohei Kobayashi, Yoshiki Yamaguchi, Yuuma Oobata, Taisuke Boku, Makito Abe, Kohji Yoshikawa, and Masayuki Umemura: Accelerating Space Radiate Transfer on FPGA using OpenCL (Accepted), International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART 2018)
Acknowledgment
This research is a part of the project titled "Development of Computing-Communication Unified Supercomputer in Next Generation" under the program of "Research and Development for Next-Generation Supercomputing Technology" by MEXT. We thank Intel University Program for providing us both of hardware and software.