Scientific Computing on Heterogeneous Architectures

Dorothea vom Bruch
Email: dorothea.vom.bruch@cern.ch

Thematic CERN School of Computing, Spring 2021
June 16th 2021
Outline

- The performance challenge
- Trade-offs between multi-core and many-core architectures
- Heterogeneous computing
- From general to specialized: Hardware accelerators and applications
- Type of workloads ideal for different accelerators
- Implications of heterogeneous hardware on the design and architecture of scientific software
- Embarrassingly parallel scientific applications in High Energy Physics
In high energy physics, usually assume flat budget for computing cost estimation
- Estimated improvement increase: 10-15% per year for the same budget
- Can no longer count on a stable increase
- Computing needs are not met
How will the trend continue?

- Smartphone market is flat
- PCs, tablets etc. sales are declining
- Little innovation incentive for companies
Trend towards heterogeneous solutions: TOP500

<table>
<thead>
<tr>
<th>Rank</th>
<th>System</th>
<th>Country</th>
<th>CPU 1</th>
<th>CPU 2</th>
<th>CPU 3</th>
<th>CPU 4</th>
<th>CPU 5</th>
<th>Power (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Supercomputer Fujikura - Supercomputer Fujikura, AIST, 400 Gbps, Dual interconnect E1, Fujitsu</td>
<td>Japan</td>
<td>IBM Power9</td>
<td>5,490</td>
<td>640</td>
<td>630</td>
<td>0.240</td>
<td>29,959</td>
</tr>
<tr>
<td>2</td>
<td>Summit - IBM Power System AC922, IBM POWER9 22C 3.77GHz, NVIDIA V100, Dual-Red Mellanox EDR Infiniband, IBM</td>
<td>United States</td>
<td>ARM processors</td>
<td>2,414,953</td>
<td>148,600</td>
<td>188,794.9</td>
<td>18,896</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Sierra - IBM Power System AC922, IBM POWER9 22C 3.19GHz, NVIDIA V100, Dual-Red Mellanox EDR Infiniband, IBM/NVIDIA/Mellanox</td>
<td>United States</td>
<td>Manycore processor: Sunway / NUDT</td>
<td>1,572,480</td>
<td>96,040</td>
<td>125,712.0</td>
<td>7,428</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Sunway Taihulight - Sunway N90, Sunway 5600/10, 256C-14.4MTH, Sunway NECPC, National Supercomputing Center in WuXi, China</td>
<td>China</td>
<td>Nvidia GPUs</td>
<td>10,649,630</td>
<td>92,014.6</td>
<td>125,620.9</td>
<td>15,371</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Jiaodao - NVIDIA DGX A100, NVIDIA EPYC 7742 v4C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, NVIDIA NVIDIA Corporation</td>
<td>United States</td>
<td>AMD &amp; Intel CPUs</td>
<td>550,320</td>
<td>62,480</td>
<td>79,275.0</td>
<td>2,046</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Tianhe-3A (HET) - FEP Cluster, Intel Xeon ES-2695V4 120 2.60GHz, Ti Express-2, Matrix-2000, NUDT, National Super Computer Center in Guangzhou, China</td>
<td>China</td>
<td></td>
<td>4,981,760</td>
<td>61,444.5</td>
<td>188,870.7</td>
<td>18,482</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>JISC/SE/MPCI - Mali Quadra M9300S, AMD EPYC 7642 2x2.9GHz, NVIDIA A100, Mellanox HDR Infiniband/Pure-Polaris Ouster/Suits, Atox Forschungszentrum Juelich/HZJ</td>
<td>Germany</td>
<td></td>
<td>443,280</td>
<td>44,010.0</td>
<td>78,980.0</td>
<td>1,764</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Aries - PowerEdge C4414, Xeon Gold 6150 2.10GHz, NVIDIA Tesla V100, Mellanox HDR Infiniband, Dell EMC, Sony</td>
<td>Italy</td>
<td></td>
<td>698,740</td>
<td>30,450.0</td>
<td>51,720.6</td>
<td>2,252</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Provena - Dell C6428, Xeon Platinum 8280 28C 2.70GHz, Mellanox Infiniband HDR, Dell EMC, Texas Advanced Computing Center/Austin, Texas</td>
<td>United States</td>
<td></td>
<td>443,448</td>
<td>23,516.4</td>
<td>38,740.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Darmstadt-7 - Cray CS-Storm, Xeon Gold 6148 26C 3.50GHz, NVIDIA Tesla V100-SMR, Infiniband HDR 100, HPE, Saudi Arabia, Saudi Arabia</td>
<td>United States</td>
<td></td>
<td>472,520</td>
<td>22,400.0</td>
<td>55,423.6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Trend towards heterogeneous solutions: TOP500

<table>
<thead>
<tr>
<th>Rank</th>
<th>System</th>
<th>Country</th>
<th>CPUs</th>
<th>GPUs</th>
<th>Manycore processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Supercomputer Fugaku</td>
<td>Japan</td>
<td>AMD &amp; Intel CPUs</td>
<td>Nvidia GPUs</td>
<td>Sunway / NUDT</td>
</tr>
<tr>
<td>2</td>
<td>Summit - IBM Power System AC45</td>
<td>Japan</td>
<td>IBM Power9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Sierra - IBM Power System AC42</td>
<td>United States</td>
<td>Nvidia GPUs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Sunway TaihuLight</td>
<td>China</td>
<td>AMD &amp; Intel CPUs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Sierra - Nvidia SX10</td>
<td>United States</td>
<td>Nvidia GPUs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Tianhe-DA - TH-MR FEP Cluster, Intel Xeon ES-2495G4</td>
<td>United States</td>
<td>Nvidia GPUs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>JNUITS/Booster Module - DU01 Sequeira F0090, AMD EPYC 7602 2.3 GHz, NVIDIA A100, Sunway NUDT</td>
<td>China</td>
<td>Nvidia GPUs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>HPCF - PowerEdge C4148, Xeon Gold 6252 2.1 GHz, NVIDIA Tesla V100, Mellanox HDR Infiniband</td>
<td>Germany</td>
<td>Nvidia GPUs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Presta - D642SE, Xeon Platinum 8280 2.7 GHz, Mellanox Infiniband HDR</td>
<td>United States</td>
<td>Nvidia GPUs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Darmstadt -7 - Cray CS-Storm, Xeon Gold 6252 2.1 GHz, NVIDIA Tesla V100 SM2, Infiniband HDR 10G</td>
<td>United States</td>
<td>Nvidia GPUs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7/10 systems are heterogeneous

https://www.top500.org/lists/top500/2020/11/
Evolution of peak FLOPs for CPUs & GPUs

- Gaming industry evolves steadily → continuous high demand for consumer GPUs
- With the trend for AI in many different areas → continuous demand for professional GPUs
What about the cost?

Heterogeneous solutions & sustainability: Green500

<table>
<thead>
<tr>
<th>Rank</th>
<th>TOP500 Rank</th>
<th>System</th>
<th>Cores</th>
<th>Rmax (TFlop/s)</th>
<th>Power (kW)</th>
<th>Power Efficiency (GFlops/watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>170</td>
<td>NVIDIA DGX SuperPOD - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz,</td>
<td>19,840</td>
<td>2,396.0</td>
<td>90</td>
<td>26.199</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>330</td>
<td>MN3 - MN-Core Server, Xeon Platinum 8260M 24C 2.4GHz, Preferred Networks</td>
<td>1,664</td>
<td>1,652.9</td>
<td>63</td>
<td>28.099</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MN-Core, MN-Core DirectConnect, Preferred Networks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Preferred Networks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japan</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>JUWELS Booster Module - Bull Sequana XH2000, AMD EPYC 7402 24C 2.8GHz,</td>
<td>449,280</td>
<td>44,120.0</td>
<td>1,764</td>
<td>26.008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NVIDIA A100, Mellanox HDR InfiniBand/ParTec ParaStation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ClusterSuite, Atos Forschungszentrum Juelich [FZJ], Germany</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>146</td>
<td>Spartan2 - Bull Sequana XH2000, AMD EPYC 7402 24C 2.8GHz, NVIDIA A100,</td>
<td>23,040</td>
<td>2,566.0</td>
<td>106</td>
<td>24.282</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mellanox HDR Infiniband, Atos Atos</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>France</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Selene - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100,</td>
<td>555,520</td>
<td>63,460.0</td>
<td>2,846</td>
<td>23.983</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mellanox HDR Infiniband, Nvidia NVIDIA Corporation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>United States</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 4/5 of the top 5 Green500 use GPUs combined with AMD Epyc
- Of the top 40 Green500:
  - 37 use accelerators
  - 2 use A64FX vector-processors
  - 1 uses a Sunway manycore processor

https://www.top500.org/lists/green500/2020/11/
GPU power efficiency

Theoretical peak FLOPs per Watt, single precision

D. vom Bruch
Multi-core versus many-core architecture

**Multi-core**
- O(10) cores
- Flexible: designed for both serial and parallel code
- Larger caches
- Emphasis on single thread performance

**Many-core**
- O(100-1000) cores
- Designed for parallel code
- Small caches
- Simpler cores

Image source
Types of workload for multi/many core architectures

- Typically, the main processor is multi-core and paired with a many-core accelerator
- Ensures that both serial and parallel code can be run efficiently
- Multi-core processors are often CPUs
  - Legacy code can run on them (albeit with low performance if not optimized for multi-threading)
  - They provide good serial performance
- Many-core processors are typically specialized accelerators
  - Individual algorithms / chains of algorithms are developed specifically for the accelerator
  - Only highly parallelizable problems are efficiently processed by them
  - The most widely used accelerators in science are many-core architectures
Heterogeneous computing

- Heterogeneous system: built of several processor types, i.e. accelerators
- Accelerators are designed to execute specific tasks
- Part of our everyday life: (de)compression, encryption, video stream decoding, 3D graphics acceleration, pattern/object recognition, automatic vehicles
- Accelerator technology often scaled to become a discrete device
  - Plug-and-play several components into a heterogeneous architecture
- Can achieve exceptional computing power with respect to CPUs with better energy efficiency if used for adequate problems

Source: https://www.iti.uni-stuttgart.de/en/chairs/ca/projects/oldprojects/simtech/
Types of hardware accelerators (used in science)

- **Graphics Processing Units (GPUs)**
  - Vendors: Nvidia, AMD
  - Intel to enter the market soon
- **Field Programmable Gate Arrays (FPGAs)**
  - Vendors: Xilinx, Altera
- **Tensor Processing Units (TPUs)**
  - Vendor: Google
- **Intelligent Processing Units (TPUs)**
  - Vendor: Graphcore
GPUs

- Developed for graphics pipeline
- General purpose computations possible
- Increasingly used for AI applications
- Hardware specialized in this direction since few years
- Programmed with high-level language

- Low core count / powerful ALU
- Complex control unit
- Large caches
  → Latency optimized

- High core count
- No complex control unit
- Small caches
  → Throughput optimized
### Types of GPUs

<table>
<thead>
<tr>
<th></th>
<th>Scientific GPUs</th>
<th>Gaming GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>~3 times more single precision TFLOPS than double precision</td>
<td>~40 times more single precision TFLOPS than double precision</td>
</tr>
<tr>
<td></td>
<td>→ suited for double precision</td>
<td>→ not well suited for double precision</td>
</tr>
<tr>
<td>Error correction</td>
<td>Available</td>
<td>Not available</td>
</tr>
<tr>
<td>Connection</td>
<td>NVLink &amp; PCIe</td>
<td>Only PCIe</td>
</tr>
<tr>
<td>Price</td>
<td>~5–6 x the price of gaming GPUs</td>
<td>Several hundred dollars Depending on model (and year)</td>
</tr>
</tbody>
</table>
# GPU vs. CPU: Specifications

<table>
<thead>
<tr>
<th></th>
<th><strong>AMD Ryzen Threadripper 3990X</strong></th>
<th><strong>Nvidia A100</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Core count</td>
<td>64 cores / 128 threads</td>
<td>6912 cores</td>
</tr>
<tr>
<td>Frequency</td>
<td>2.9 GHz</td>
<td>1.41 GHz</td>
</tr>
<tr>
<td>Peak Compute Performance</td>
<td>3.7 TFLOPs</td>
<td>19.5 TFLOPs (single precision)</td>
</tr>
<tr>
<td>Memory bandwidth</td>
<td>Max. 95 GB/s</td>
<td>1.6 TB/s</td>
</tr>
<tr>
<td>Memory capacity</td>
<td>Max O(1) TB</td>
<td>40/80 GB</td>
</tr>
<tr>
<td>Technology</td>
<td>7 nm</td>
<td>7 nm</td>
</tr>
<tr>
<td>Die size</td>
<td>717 mm(^2)</td>
<td>826 mm(^2)</td>
</tr>
<tr>
<td>Transistor count</td>
<td>3.8 billion</td>
<td>54.2 billion</td>
</tr>
<tr>
<td>Model</td>
<td>Minimize latency</td>
<td>Hide latency through parallelism</td>
</tr>
</tbody>
</table>
## Connectivity with GPU: PCIe connection

<table>
<thead>
<tr>
<th>PCIe generation</th>
<th>1 lane</th>
<th>16 lanes</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>500 MB/s</td>
<td>8 GB/s</td>
<td>2007</td>
</tr>
<tr>
<td>3.0</td>
<td>985 MB/s</td>
<td>15.75 GB/s</td>
<td>2010</td>
</tr>
<tr>
<td>4.0</td>
<td>1.97 GB/s</td>
<td>31.5 GB/s</td>
<td>2017</td>
</tr>
</tbody>
</table>
FPGAs

- Thousands of logic blocks
- I/O blocks
- Connected via programmable interconnect

- Configure a circuit to do the task it is programmed for → Hardware implementation of an algorithm
- Fixed latency
- Very good at integer computations
- Does not require a computer to run (has its own I/O)
- Traditionally, programmed with hardware description languages (Verilog, VHDL) → long development time
- Increasingly more high-level languages developed

Source: National Instruments
GPU vs. FPGA

**GPUs**
- Higher latency
- Connection via PCIe (or NVLink)
- Bandwidth limited by PCIe
- Very good floating point operation performance
- Lower engineering cost
- Backward / forward compatibility

**FPGAs**
- Low & deterministic latency
- Connectivity to any data source
- High bandwidth
- Intermediate floating point performance
- High engineering cost
- Not so easy backward compatibility
# CPU – GPU - FPGA

<table>
<thead>
<tr>
<th></th>
<th>Latency</th>
<th>Connection</th>
<th>Engineering cost</th>
<th>FP performance</th>
<th>Serial / parallel</th>
<th>Memory</th>
<th>Backward compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>O(10) μs</td>
<td>Ethernet, USB, PCIe</td>
<td>Low entry level: Programmable with C++, python, etc.</td>
<td>O(1-10) TFLOPs</td>
<td>Optimized for serial, increasingly vector processing</td>
<td>O(100) GB RAM</td>
<td>Compatible, except for vector instruction sets</td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td>O(100) μs</td>
<td>PCIe, Nvlink</td>
<td>Low to medium entry level: Programmable with CUDA, OpenCL, etc.</td>
<td>O(10) TFLOPs</td>
<td>Optimized for parallel performance</td>
<td>O(10) GB</td>
<td>Compatible, except for specific features</td>
</tr>
<tr>
<td><strong>FPGA</strong></td>
<td>Fixed O(100) ns</td>
<td>Any connection via PCB</td>
<td>High entry level: traditionally hardware description languages, Some high-level syntax available</td>
<td>Optimized for fixed point performance</td>
<td>Optimized for parallel performance</td>
<td>O(10) MB on the FPGA itself</td>
<td>Not easily backward compatible</td>
</tr>
</tbody>
</table>

Developed for AI applications: TPUs & IPUs

Google’s TPUs:
- ASIC for neural network machine learning (TensorFlow)
- Designed for high amount of low precision matrix manipulation
- Available in Google Cloud
- Can fit in hard drive slot within a data center rack
- Could be used for training / inference within HEP: reference

Graphcore’s IPUs:
- Developed for machine learning applications
- Multiple Instruction Multiple Data (MIMD) paradigm
- For irregular and sparse data access
- Individual threads run on small data blocks
- Application in HEP reconstruction: reference
Types of workloads for different accelerators

**GPUs:**
- Relaxed latency requirements
- High FLOPs need
- I/O via PCIe no bottleneck
- Highly parallelizable problem
- Fits within GPU memory

**FPGAs:**
- Strict latency requirements
- High I/O needs
- Highly parallelizable problem
- Fits within FPGA resources (logic elements and memory blocks)

**TPUs / IPUs etc.:**
- Machine learning training or inference
- TPUs: Use as a service in the cloud
- IPUs: MIMD compatible problem
- Fit within memory
Challenges in heterogeneous computing

• Different architectures
  • Different instruction sets can produce results that are not bit-wise reproducible

• Data transmission between devices
  • Interconnect: bandwidth bottleneck
  • Data layout: one might not be suitable for all device architectures and memory structure

• Programming environments
  • Different compilers
  • Different APIs
Challenges in heterogeneous computing

- Different architectures
  - Different instruction sets can produce results that are not bit-wise reproducible

- Data transmission between devices
  - Interconnect: bandwidth bottleneck
  - Data layout: one might not be suitable for all device architectures and memory structure

- Programming environments
  - Different compilers
  - Different APIs

Check requirements of problem at hand:
What is the minimum required resolution?

Minimize copies between devices
Minimize transformations between data layouts

Port code to run on accelerators
Use programming environments designed for heterogeneous computing
→ lecture on Friday
Computing needs in HEP

- Detector
- Trigger / Real-time analysis
- Storage
- Simulation
- Data analysis
Real-time analysis / Trigger

Local characteristic signature, For example high energy / pt particle

Analysis of whole event required to select events → reconstruct all trajectories
Event selection locally in hardware

- Necessary if not the whole data stream can be read out
- Possible if decision is based on local objects
- For example: part of a calorimeter
- Need low latency and high bandwidth \(\rightarrow\) optimal for FPGAs / ASICS
  \(\rightarrow\) hardware level triggers
Event selection globally in software

- Only possible if the whole data stream can be read out (or has been reduced before by a hardware level trigger)
- Necessary if the information from all (or several) detectors is needed for the selection decision
- No hard latency requirement
- Input data provided from servers that gather information of one proton bunch collision from all sub-detectors ("event building")
- "High Level Trigger" (HLT)
- Optimal task for heterogeneous architectures
Challenge I: Real-time analysis (RTA)

- Run 3:
  - ALICE: PbPb collisions at 50 kHz (pileup of 6) → 3.5 TB/s
  - LHCb: pp collisions at 30 MHz → 5 TB/s

- Run 4:
  - CMS & ATLAS will collect pp collision data with a pileup of 200
  - Increase hardware trigger rate from 100 kHz to 750-1000 kHz → around 6 TB/s processed in software

- Run 5:
  - LHCb undergoes second upgrade → 40 TB/s in software
Heterogeneous solutions for RTA

• ALICE has been using a heterogeneous system of CPUs and GPUs for the high level trigger since Run 1
• This scheme is kept for the Run 3 upgrade
• LHCb will start using a heterogeneous system based on CPUs and GPUs in Run 3
• CMS is exploring a heterogeneous system for Run 3, test run already planned for Run 3
• Smaller experiments also moving in this direction: Mu3e, NA62

• All experiment needs and environments are quite different
• → heterogeneous solutions are different

Commonalities:
• Reconstruction algorithms are main candidates for parallelization and off-loading to accelerators
• Scheduling of memory copies, calculations on accelerator, calculations on host server is crucial
• Flexible software frameworks are necessary
Recurrent tasks in real-time data analysis

**Raw data decoding**
- Transform binary payload from subdetector raw banks into collections of hits \((x,y,z)\) in LHCb coordinate system

**Track reconstruction**
- Consists of two steps:
  - Pattern recognition: Which hits were produced by the same particle? → “Track”
    - Huge combinatorics when testing different combinations of hits
  - Track fitting: Describe track with mathematical model

**Vertex finding**
- Where did proton-proton collisions take place?
- Where did particles decay within the detector volume?

**Calorimeter / muon detector reconstruction**
- Reconstruct clusters in the calorimeter / muon detectors
- Match tracks to clusters
Recurrent tasks in real-time data analysis

Raw data decoding
- Transform binary payload from subdetector raw banks into collections of hits \((x,y,z)\) in LHCb coordinate system

Track reconstruction
- Consists of two steps:
  - Pattern recognition: Which hits were produced by the same particle? \rightarrow \text{“Track”}
    \rightarrow \text{Huge combinatorics when testing different combinations of hits}
  - Track fitting: Describe track with mathematical model

Vertex finding
- Where did proton-proton collisions take place?
- Where did particles decay within the detector volume?

Calorimeter / muon detector reconstruction
- Reconstruct clusters in the calorimeter / muon detectors
- Match tracks to clusters

Track reconstruction: Most compute intensive part
- Many computations on independent data
- Can process combinations of hits in parallel
- Can process track candidates in parallel
- \rightarrow well suited for many-core architectures
ALICE experiment

- One of the four large experiments at the LHC
- Take data with proton and heavy ion collisions
- Study quark-gluon plasma
- What happened just after the big bang?
- Main detector: large time projection chamber (TPC)
ALICE: Reconstruction on GPUs

- Process 10 ms timeframes, O(10 GB) size
- One detector dominates computing needs: Time Projection Chamber (TPC)
- TPC reconstructed in real time on GPUs for compression and calibration since Run 1
- Also adding reconstruction of other detectors to the GPU workflow
- Aiming to process full barrel reconstruction on GPUs
- New facility for data processing and compression – 1500 CPU/GPU nodes, 60 PB storage

See D. Rohr’s vCHEP talk
ALICE TPC reconstruction on GPUs

- Run several events in parallel
- The event size is large, so not too many events fit into GPU memory at once
- Process the sectors of the TPC in parallel
- Same code base for CPU and GPU code → can run on either architecture

<table>
<thead>
<tr>
<th>#</th>
<th>Phase</th>
<th>Task</th>
<th>Method</th>
<th>Locality</th>
<th>Time</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I</td>
<td>Seeding</td>
<td>Cellular Automaton</td>
<td>Very local</td>
<td>30 %</td>
<td>CPU &amp; GPU</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Track following</td>
<td>Simple Kalman filter</td>
<td>Sector-local</td>
<td>60 %</td>
<td>CPU &amp; GPU</td>
</tr>
<tr>
<td>3</td>
<td>II</td>
<td>Track Merging</td>
<td>Matching Covariance</td>
<td>Global</td>
<td>2 %</td>
<td>CPU</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Final Fit</td>
<td>Kalman filter</td>
<td>Global</td>
<td>8 %</td>
<td>CPU (or GPU)</td>
</tr>
</tbody>
</table>

D. vom Bruch

arxiv 1712.09430
LHCb experiment

- One of the four large LHC experiments
- Mainly specialized in the physics of beauty hadrons
- Single-arm forward spectrometer
- Run 3: upgrade many detectors and full readout
LHCb: GPUs used for first data selection

Raw data of one event ~100 kB
→ not limited by PCIe connection
→ can fit many events into GPU memory at once
LHCb High Level Trigger 1

- Decode binary payload of four sub-detectors
- Reconstruct charged particle trajectories
- Identify muons
- Reconstruct primary and secondary decay vertices
- Select pp-bunch collisions based on
  - Single-track properties
  - Secondary vertex properties

- Manageable amount of algorithms with highly parallelizable tasks
- Ideally suited for GPUs
LHCb HLT1 computing performance

- Full HLT1 sequence implemented on GPUs
- Require about 215 GPU cards to process HLT1 @ 30 MHz
- Have slots for 500 cards
- Significant throughput increase on latest generation Nvidia GPUs
History: HLT1 architecture choice

Proposal in TDR (2014)  
CERN-LHCC-2014-016

- pp collisions
  - 170 servers
  - event building
  - 40 Tbit/s
  - 30 MHz
  - HLT1
  - buffer on disk calibration and alignment
  - 80 Gbit/s
  - storage

Updated strategy (as of 5/2020)  
CERN-LHCC-2020-006

- pp collisions
  - 170 servers
  - event building
  - 40 Tbit/s
  - 30 MHz
  - GPUs
  - HLT1
  - buffer on disk calibration and alignment
  - 1-2 Tbit/s
  - ~1 MHz
  - HLT2
  - storage

- Developed two solutions simultaneously
- Both the multi-threaded CPU & the GPU HLT1 fulfilled the requirements from the 2014 TDR
- Detailed cost benefit analysis (arXiv:2105.04031)
- GPU solution leads to cost savings on processors and the network
- Throughput headroom for additional features
- Decision: A GPU-based software trigger will allow LHCb to expand its physics reach in Run 3 and beyond.

See also arXiv:2106.07701 on LHCb’s energy efficiency with a CPU and GPU HLT1
ATLAS & CMS experiments

- Not one single subdetector dominating workload
- More subdetectors compared to LHCb and ALICE
  → more possibilities for paths a track can take, more complex geometry
- Event size smaller than ALICE, larger than LHCb
- But huge computing challenge → exploring heterogeneous solutions

- ATLAS GPU R&D inside ACTS framework (“A Common Tracking Software”)
- Generic detector geometry (similar to ATLAS / CMS)
- Designed to develop and compare track reconstruction algorithms
- Few pattern recognition and track fitting algorithms implemented for GPUs

Speedup of pattern recognition algorithm on GPU (RTX 1070) vs CPU (i7-5820K)

CMS reconstruction on GPUs

- Several algorithms ported to GPUs for Run 3:
  - Track reconstruction in pixel detector
  - Primary vertex reconstruction from those tracks
  - Calorimeter local reconstruction
  - Out-of-time pileup subtraction
- Crucial to allow close interlinking of CPU and GPU software
  → integrated into CMSSW (arXiv2004.04334)
- Work ongoing for other reconstruction algorithms
CMS HLT performance with GPUs

- GPU offload provides 25% increase for HLT throughput
- GPU candidate: Tesla T4
- 25% lower cost
- 20% less power consumption

See LHCP talk
Mu3e experiment

- Fixed target experiment at the Paul Scherrer Institute in Switzerland
- Study lepton flavor violating decay $\mu^+ \rightarrow e^+ e^- e^+$
- Triggerless readout @ 10 GB/s, reduce to 100 MB/s with GPU filter farm
- Process 50 ns time slices of data
- Linear track fit for low-momentum particles for real-time data selection implemented on GPUs
- Measured $2 \cdot 10^6$ time slices / s on one Nvidia GTX 1080

$\rightarrow$ Can do full event selection with 12 GPUs

- Planned to start data-taking in 2023

EPJ Web of conferences, 2017
## Overview of GPU usage in various HEP experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Main tasks processed on GPU</th>
<th>Event / data rate</th>
<th>Number of GPUs</th>
<th>Deployment date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mu3e</td>
<td>Track- &amp; vertex reconstruction</td>
<td>20 MHz / 32 Gbit/s</td>
<td>O(10)</td>
<td>2023</td>
</tr>
<tr>
<td>CMS</td>
<td>Decoding, clustering, pattern recognition in pixel detector</td>
<td>100 kHz</td>
<td></td>
<td>2022 (tbc)</td>
</tr>
<tr>
<td>ALICE</td>
<td>Track reconstruction in three sub-detectors</td>
<td>50 kHz Pb-Pb or &lt; 5 MHz p-p / 30 Tbit/s</td>
<td>O(2000)</td>
<td>2022</td>
</tr>
<tr>
<td>LHCb</td>
<td>Decoding, clustering, track reconstruction in three sub-detectors, vertex reconstruction, muon ID, selections</td>
<td>30 MHz / 40 Tbit/s</td>
<td>O(250)</td>
<td>2022</td>
</tr>
</tbody>
</table>


D. vom Bruch
Common characteristics of software frameworks

- Same code base compiled for various computing architectures: GPUs, x86,…
- Memory management system for GPU memory: avoid dynamic memory allocation
- Schedule pipelines of GPU (and CPU) algorithms → hide memory copies
- Integration into experiments’ main software frameworks

Allen framework at LHCb
Patatrack at CMS
O2 at ALICE
Challenge II: Simulation

- Running experiments at higher luminosity leads to large increase in simulation demands
- Projected between 45 and 90% of CPU usage for simulation
- Large effort ongoing to process simulation on GPUs
- Partially driven by hardware available in HPC centers

Projected needs for Run 3

See H. Gray’s opening vCHEP talk
Event generators on GPUs

- Madgraph4gpu project: started in 2020 within HSF Generator WG
- Port MC event generators, in particular matrix element calculation (current bottleneck), to GPUs
- Make use of CUDA’s random number generator: cuRAND
- Discussion: Is floating point precision enough?

Main design idea: event-level data parallelism (lockstep)

- In MC generators, all events in one channel initially go through the same calculations
  - Computing MEs involves the calculation of the exact same function on different data points
  - This is what makes event generators a good fit for GPUs (SIMT) and vector CPUs (SIMD)

Event-level parallelism in practice – coding and #events

- Easier to code for GPU SIMT than for CPU SIMD: CUDA code was faster to prototype
- CUDA (GPU) implementation
  - For SIMT, event loop is “orthogonal”: one thread = one event (GPU thread ID = event ID)
  - For SIMT, SOA memory layouts are beneficial (coalesced access), but not strictly essential
- C++ (CPU) implementation
  - For SIMD, event loop must be the innermost loop (e.g. invert helicity and event loops)
  - For SIMD, SOA memory layouts in the computational kernel are essential
- To be efficient, CUDA needs O(10k)-O(1M) events in parallel – much more than C++!
  - CUDA: lockstep within each warp (32 threads) + many warps in parallel to fill the GPU
  - C++: lockstep within a vector register (2-8 doubles) + multi-threading or multi-processing
Photon simulation with Nvidia OptiX

- Photon simulation is similar to ray tracing problem
  → ideally suited workload for GPU
- Opticks framework developed for photon simulation, e.g. in a LAr TPC
- Uses Nvidia’s OptiX ray tracing engine and integrated with Geant4

See S. Blyth’s vCHEP talk

Also IceCube are working on using ray tracing for their photon simulation, see this vCHEP talk
R&D to use Graphcore’s IPUs

**Simulation**
- Study usage of IPUs for event generation with fast simulation technique
- Particularly suited for machine learning techniques
- Tested event generation with generative networks (GAN)

**Track reconstruction**
- Also implemented Kalman filter for track fitting on the IPU
- Multiple Instruction Multiple Data (MIMD) architecture
- Higher performance observed for conditional control-flow programs
- No direct comparison to GPU implementation

![Fig. 3: Comparison of the time to train the IPU relative to the CPU or GPU of Table 1](image-url)

Computing and Software for Big Science 5, 8 (2021)
Summary

- We are facing a huge computing challenge in HEP, mainly in real-time reconstruction and simulation
- Cannot be solved solely by using CPU processors
- Trend in HPC is towards heterogeneous architectures
- Make use of many-core accelerators for embarrassingly parallel problems within HEP
- Most popular accelerator: GPUs (most theoretical TFLOPs/$, power efficient)
- Various experiments plan to use GPUs for software triggers in the coming years
- Extensive R&D also ongoing to use them for simulation
- Frameworks for heterogeneous software are being developed
- DAQ systems are adopted to accommodate accelerators

- Note: Compute Accelerator Forum organized by HSF, Openlab, SIDIS
  Presentations roughly once per month on accelerator topics
Backup
Cellular Automaton

- Build short track segments
- Find segments from different layers that fit together
  → build tree structure
- Track: tree with minimum number of connected segments
- Data locality and intrinsically parallel → well suited for many-core architecture
Kalman Filter

- One method for track fitting
- Subsequently iterates over all hits on a track
- For every hit, estimate the state of the track at that location:
  - First: predict it based on the previous state
  - Second: update it based on the measurement (hit)

Arbitrary initial state vector

D. vom Bruch
Kalman Filter

• One method for track fitting
• Subsequently iterates over all hits on a track
• For every hit, estimate the state of the track at that location:
  • First: predict it based on the previous state
  • Second: update it based on the measurement (hit)
Kalman Filter

- One method for track fitting
- Subsequently iterates over all hits on a track
- For every hit, estimate the state of the track at that location:
  - First: predict it based on the previous state
  - Second: update it based on the measurement (hit)
Kalman Filter

• One method for track fitting
• Subsequently iterates over all hits on a track
• For every hit, estimate the state of the track at that location:
  • First: predict it based on the previous state
  • Second: update it based on the measurement (hit)
Kalman Filter

- One method for track fitting
- Subsequently iterates over all hits on a track
- For every hit, estimate the state of the track at that location:
  - First: predict it based on the previous state
  - Second: update it based on the measurement (hit)
- At last plane: best linear estimator for track state
- Only parallelizable over all tracks in one event
GPUs for data analysis

Example

- Partial wave analysis using GooFit
- GooFit: GPU-friendly framework for maximum-likelihood fits
- Backend for Nvidia GPUs (using Thrust library) and for CPUs using OpenMP

Data analysis

Table 1. GPU performance of the MIPWA fit.

<table>
<thead>
<tr>
<th>Platform</th>
<th>GPU Model</th>
<th>Chip</th>
<th>CUDA cores</th>
<th>Run time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tWorkstation</td>
<td>Tesla K40c</td>
<td>GK110BGL</td>
<td>2880</td>
<td>76</td>
</tr>
<tr>
<td>Desktop PC</td>
<td>GeForce GTX 980</td>
<td>2nd gen. Maxwell (GM204)</td>
<td>2048</td>
<td>67</td>
</tr>
<tr>
<td>Laptop ASUS N56V</td>
<td>GeForce GT 650M</td>
<td>GK107</td>
<td>384</td>
<td>179</td>
</tr>
</tbody>
</table>

Running on one CPU core: 8 hours

arxiv.org 1703.03284
### Some Nvidia GPUs

<table>
<thead>
<tr>
<th>Feature</th>
<th>GeForce GTX 1060</th>
<th>GeForce GTX 1080 Ti</th>
<th>GeForce GTX 2080 Ti</th>
<th>Tesla T4</th>
<th>Tesla V100</th>
</tr>
</thead>
<tbody>
<tr>
<td># cores</td>
<td>1280</td>
<td>3584</td>
<td>4352</td>
<td>2560</td>
<td>5120</td>
</tr>
<tr>
<td>Max. frequency</td>
<td>1.81 GHz</td>
<td>1.67 GHz</td>
<td>1.55 GHz</td>
<td>1.59 GHz</td>
<td>1.37 GHz</td>
</tr>
<tr>
<td>Cache (L2)</td>
<td>1.5 MB</td>
<td>2.75 MB</td>
<td>6 MB</td>
<td>6 MB</td>
<td>6 MB</td>
</tr>
<tr>
<td>DRAM</td>
<td>5.94 GB GDDR5</td>
<td>10.92 GB GDDR5</td>
<td>10.92 GB GDDR5</td>
<td>16 GB GDDR6</td>
<td>32 GB HBM2</td>
</tr>
</tbody>
</table>

**Gaming GPUs**

**Scientific GPUs**