Scaling Data Processing from CPU to distributed GPU – William Malpica
Summary notes

Introduction:

- The talk focuses on scaling data processing from CPUs to distributed GPUs.
- CPUs are good at everything but GPUs excel at parallel processing of structured data.

CPU vs GPU:

- CPUs are generic, easy to use, low latency, efficient at different tasks (like USPS trucks).
- GPUs are better at doing the same task repeatedly in parallel, prefer structured data in vectors/arrays, high latency but high throughput (like shipping trucks).

Data Processing Types:

- OLTP (Online Transactional Processing): Row-oriented databases like MySQL, PostgreSQL; row-wise file formats like CSV, TFRecord. Not GPU-friendly.
- OLAP (Online Analytical Processing): Column-oriented databases like Snowflake, Redshift; column-wise file formats like Apache Parquet, Apache ORC. GPU-friendly.

Apache Arrow:

- Columnar memory format, language-independent, efficient for analytics on modern CPUs and GPUs.
- Supports zero-copy data transfers, de-facto standard in modern data analytics.

GPU Acceleration Examples:

- cuDF (like Pandas) is orders of magnitude faster than CPU for operations like GroupBy, Sort, Merge.
- CuPy (like NumPy) is faster for array operations like FFT, matrix multiplication, SVD.
- cuML (like scikit-learn) is faster for machine learning algorithms like PCA, regression, clustering.
- cuGraph (like NetworkX) is significantly faster for graph analytics.

Other GPU-accelerated Libraries:

- PyTorch, ArrayFire, TensorFlow, Numpy, Bend, XGBoost, Voltron Data Theseus SQL Engine.

Why GPUs are Fast:

- More cores (orders of magnitude more than CPUs).
- Higher memory bandwidth (20-30x faster than CPU memory).

When to Use GPUs:

- Use GPUs when the workload is compute-bound, compute density is important, throughput speed enables doing more, or you're already using GPUs.
- CPUs might be better for latency-bound, I/O-bound workloads, when costs outweigh benefits, or when data is not too large.

GPU-Powered Workflows:

- Can reduce iteration time for data scientists by accelerating ETL and analysis steps.

Scaling to Distributed GPUs:

- For large data problems, distributed solutions are required.
- GPUs can help overcome the performance wall faced by CPUs when scaling out.
- New bottlenecks arise: networking and storage.

Hardware Accelerators:

- PCIe bandwidth for storage, NVLink bandwidth for networking.
- Technologies like GPUDirect RDMA, GPUDirect Storage can accelerate data transfer.

Distributed GPU Systems:

- Dask with OpenUCX for GPUDirect RDMA.
- Spark Rapids for GPUDirect RDMA and GPUDirect Storage.
- MPI with GPUDirect RDMA.
- Voltron Data Theseus for GPUDirect RDMA and GPUDirect Storage.

Voltron Data Theseus:

- Accelerator-native distributed query engine.
- Petabyte-scale workloads.
- Composable, built on open standards.
- Evolutionary, adapts to new hardware and languages.