Data Centric Systems Approach
Technology directions …

- The future is Exascale
- Solutions requirements are increasingly complex
- Data requirements are exploding
- But technology is slowing

I/O Performance / Capacity losing ground …

Microprocessor clock rates have stalled …

And network bandwidth can’t keep up.
Data Centric Systems – High Level View

- Analytics
- Visualization Interpretation
- Data
- Modeling Simulation
- Data Acquisition

Massive Scale Data and Compute
IBM’s Data Centric Systems Approach

- Data is Central
  - Architect a complete system solution which addresses data

- Compute where ever there is data.

- Move data only when absolutely essential

- To manage complete workflows within power and cost constraints, heterogeneity is essential
  - so accept and embrace.

- Programming models have to be enabling and have to support performance portable code.
Heterogeneity – The critical element to address real workloads

Analytics Capability:
- Complex code
- Data Dependent Code Paths / Computation
- Lots of indirection / pointer chasing
- Often Memory System Latency Dependent
- C++ templated codes
- Limited opportunity for vectorization
- Limited scalability
- Limited threading opportunity

Massively Parallel Compute Capability:
- Simple kernels,
- Often ops dominated (e.g. DGEMM, Linpack)
- Simple data access patterns.
- Can be preplanned for high performance.

Reservoir
4-40 Racks

Oil and Gas

Seismic

Financial Analytics
2-20 Racks

All Source Analytics
1-5+ Racks

Graph Analytics
Science
1-100’s Racks

Throughput

Value At Risk

Image Analysis
Capability
Optimized System Design for Data Centric Systems

- Integration of massive data management and compute with complex analytics
- Optimized workflow components (compute and dataflow) across the system
- Data centric systems move computation to the data
  - Software provides middleware, programming models, APIs and workflow optimization tools
So where are we - CORAL 2017

- At least 5X Titan / Sequoia Application Performance
- Approximately 3,400 nodes, each with:
  - Multiple IBM POWER9™ CPUs
  - Multiple NVIDIA Tesla® GPUs using the NVIDIA Volta architecture
  - CPUs and GPUs completely connected with high speed NVLink
  - Large coherent memory: over 512 GB (HBM + DDR4)
  - all memory directly addressable from the CPUs and GPUs
- Over 40 TF peak performance per node
- Dual-rail Mellanox® EDR-IB full, non-blocking fat-tree interconnect
- IBM Elastic Storage (GPFS™) - 1TB/s I/O and 120 PB disk capacity.

- Approaching 1 Petaflop in a Rack
- With this specific technology we could assemble a usable 500 PetaFlop system in 2018
To address complex workload computational requirements within cost and power constraints, design space is limited.

The node is now a workflow and engineering optimization problem.
Multiple compute engines
- Consider all engines as equal peers

Multiple memories
- Consider all memories as equal peers
Programming Approaches

- **Accelerator approach (Copy-only model):**
  - Required when not coherent
  - Each processor computes in its own private address space
  - Data objects are stored in CPU memory and copied to GPU memory for GPU execution; GPU engines act only on data in GPU memory.

- **Compute in shared address space**
  - New option, now that CPU / GPU memories are coherent
  - Data objects can be in any physical memory domain
  - Processors (either CPU or GPU) can use data in place
  - No copies required

- **Note:**
  - Will still have to manage NUMA

- **GOAL:** Performance portable across multiple architectures
- **Target approach:** MPI (PGAS) + OpenMP 4.x using C, C++, Fortran
Peer Processing

- Data can be placed at Allocation, or can Migrate under run time control (e.g. UVM)
- Thread-like programming model (OpenMP 4.x, OpenACC, CUDA …)
- Still under development …
Challenge 1: Understand/Manage Data Organization and Flow

- Data will be complex.
- Data will be multifaceted
- Workflows will evolve data sets
  - Computation will move to the data
  - User will control placement of work not just to compute nodes, but to IO nodes, storage etc.
  - Data will flow through the system only when
- Critical to understand / plan up front
Challenge 2: Exposing Thread Level Parallelism

- Expect large numbers of compute thread engines on a node
  - MPI only not reasonable

- Need to identify / expose parallel opportunities in applications
  - Needs to be flexible
    - e.g. 13 threads per MPI on system 1, 29 threads per MPI on system 2
  - Code should be agnostic to specific memory and CPU architecture
  - Data location and execution location should be a runtime selection of user
Challenge 3: Managing Memory Pools / Affinity

- Expect multiple memory pools within a single shared memory domain
  - Performance will depend critically on proper affinity of data with compute

- Need to structure codes to allow easy allocation of data within different memory pools
  - Plan data structures and allocations carefully
  - Allocations should be discrete, robust
  - Allocate / DeAllocate only when needed?
  - Should be selectable at runtime
Commercial Programming Model: Hadoop / SPARK

- Big data workflow is typically a combination of common workload components
- Classic Database and NoSQL data
  - Querying of structured or semi-structured data
  - NoSQL data systems: store, index, query (e.g. on Tweets)
- Hard core commercial
  - Collaborative Filtering, clustering, web search, recommendation, ...
- Streaming
  - Ad and social network monitoring, real-time analysis of machine data, fraud and anomaly detections, ...
- Pleasingly parallel (local analytics)
  - Sensor data filtering, classification, statistical averages/histogram (e.g. LHC, Astronomy, Pathology, Bioimaging, ...)
- Global analytics/global machine learning
  - Deep Learning, Multidimensional Scaling, Graph Community finding to Shortest Path, ...

![Diagram of Apache Hadoop Ecosystem](image)
Data Centric Systems Programming Models

HPC Evolutionary Path (POR)
- OpenMP + X (MPI, PGAS, …)
- CORAL solution,
  - must continue to be viable beyond CORAL to Exascale
- NUMA will drive significant evolutions
  - Data and compute affinity

Analytics and Commercial
- Spark:
  - Distributed Data Objects
  - Implicit Parallelism

Approach:
- Peer Processing within a coherent, shared address space
- Evolve MPI, PGAS, …, SPARK approaches to provide high performance, low latency code implementations supporting small messaging on distributed data objects
Summary

- Data is now the first class citizen in architecture design for emerging commercial and Exascale systems.
- Compute everywhere is essential to address the bandwidth wall.
- Heterogeneity in memory and compute is a given within cost / power constraints (not going away)
  - Expect even more heterogeneity in future,
  - FPGAs, Quantum?, other
- Programming models and system software must embrace and accept “Data First” concept.
- For better or worse, programming models are already mostly defined and evolving from two directions:
  - OpenMP + X (e.g. MPI, PGAS, …)
  - HADOOP / SPARK