The Intersection of Big Data and HPC:
Using Asynchronous Many Task Runtime Systems for HPC & Big Data

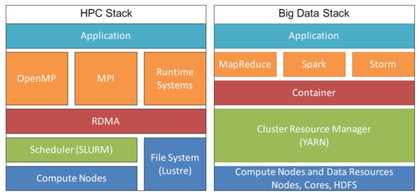
Joshua Suetterlein, Joshua Landwehr, Andres Marquez, Joseph Manzano, Kevin J Barker and Guang R. Gao

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

Overview

- Promises of agile computing might help to leverage the HPC / Big Data Chasm
  - Different hardware substrate with different requirements
  - Commodity versus custom networks
- Different type of accelerated hardware
  - Workloads with different characteristics and constrains
    - Precision, data volume, access patterns, Locality, etc
- Exploratory Vehicles: Asynchronous Many Task Runtime Systems (AMT)
  - Smaller units of work
  - Reconfigurable schedulers & data layout distributions
  - Massive concurrency exploitation



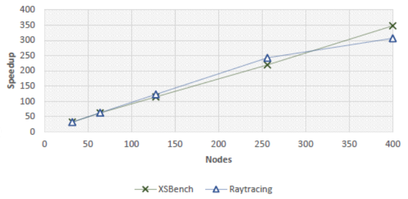
Example AMTs: HPX, ArgoBots, Legion, OCR

The Open Community Runtime

Computation expressed as DAGs in which nodes are abstractions for computations (called ***Event Driven Tasks***), data (referred as ***DataBlocks***) and synchronization (a.k.a. ***Events***), visible to the entire computational system

Unique features:

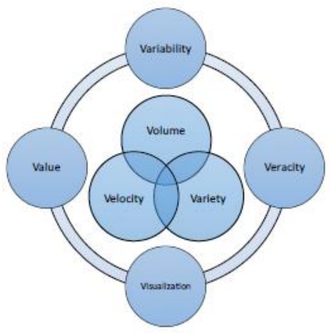
- Locality hints, aggregation and coalescing of internode messaging
- Out of order engine for outstanding requests
- Support for TCP over verbs, MPI and TCP/IP
- Advance coherency and consistency memory at scale
- Model based adaption
- Support for HPC and Big Data Workloads



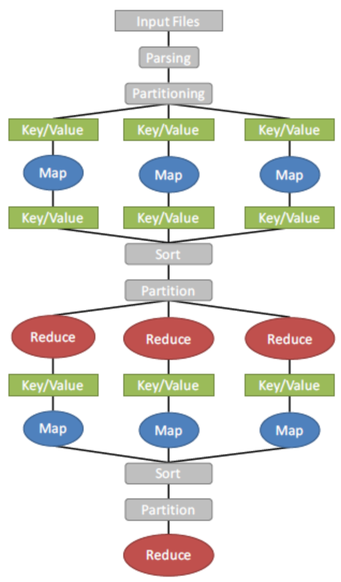
Strong Scaling Results for XSBench and RayTracing on 400 Intel based nodes

Big Data Extensions

- Big data primitives: Key / Value Pairs, Big Data partitions & operators
  - Partitions
    - Composed of key/value pairs
    - Basic abstraction for concurrency
  - Operators
    - Mappers, shufflers, sorters, etc (based on the MapReduce execution model)



Big data Properties

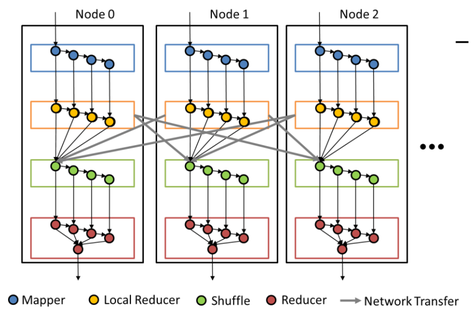


A Typical MapReduce Program

Big Data Extensions: Mapping to OCR

- P-OCR Extensions for Big Data***: Iterable DataBlocks & Big Data Streaming Operators***
- Iterable DataBlocks: K/V pairs containers with distinct semantics and specific trade-offs between time and space
  - **Bi-directional:** Linked list based datablock in which K/V can be retrieved in an order (forward or backward)
  - **Random Access:** Array based datablock in which K/V pairs can be retrieved by indexing
  - **Hash:** Hash based datablock in which K/V are retrieved based on a key
  - **Red Black Tree:** A sorted repository of K/V pairs
- Big Data Streaming Operators: Mappers, Shufflers, Reducer & Local Reducer
  - Form a computational chain with iterable datablocks as inputs
  - **Mappers:** Parsing, transforming or filtering K/V pairs from the file system
  - **Shufflers:** Partitioning or aggregating K/V pairs per node
  - **Reducer:** Broad operations across an entire partition
  - **Local Reducer:** Perform an operation on an iterable DataBlock prior of sending data through the network.

Concurrency Model for Operators



– **Big Data Computational chains' Parallelism:**
  - *Inter Chain:* There may be multiple instances of the chain running in parallel
  - *Intra Chain:* A single chain component may decide to "grow" if a threshold is triggered

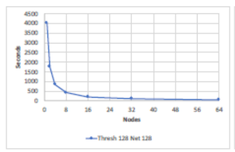
– Semantically, certain operators serve as a sync point for the spawned operators of this chain

Experimental Testbed

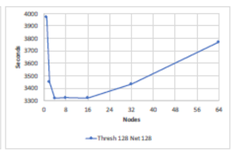
Testbeds

Hardware Testbed : 128 nodes (32-core AMD BullDozers), FDR InfiniBand LUSTRE File system.
Workloads: Word Count with up to 2 Terabyte data sets, TeraSort with up to half a Terabyte
Testing Methodology: Strong and weak scaling studies with real and synthetic workloads
DataBlock Size: Experiments using both best empirical block size versus adaptive runtime selection of block size starting at 1

Word Count Non Adaptive Results



Strong scaling 32 GB Total

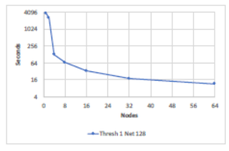


Weak scaling 32 GB per node

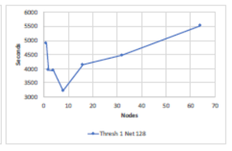
Nodes	Strong	Weak
1	1	1
2	2.29	1.15
4	4.76	1.19
8	9.64	1.19
16	19.36	1.19
32	36.86	1.15
64	66.74	1.05

Relative Speedup

TeraSort Non Adaptive Results



Strong scaling 14.5 GB Total



Weak scaling 7.3 GB per node

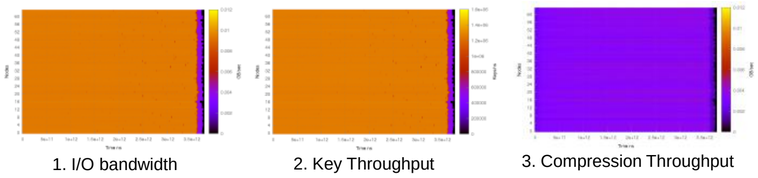
Nodes	Strong	Weak
1	1	1
2	1.46	1.23
4	28.39	1.24
8	55.42	1.52
16	107.74	1.18
32	201.36	1.09
64	310.75	0.88

Relative Speedup

Word Count Case Study

In-Depth Application Analysis for Weak Scaling

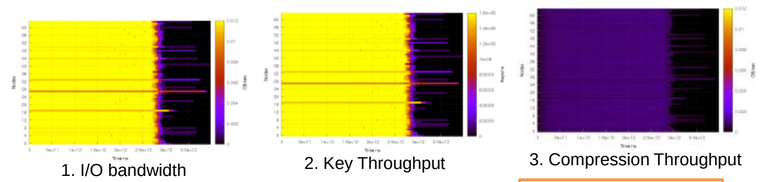
Non adaptive Runtime Behavior for Word Count Running on 64 Nodes



1. I/O bandwidth
2. Key Throughput
3. Compression Throughput
4. Network Bandwidth
5. Reducer Throughput

The ***network*** presents the biggest performance bottleneck as ***compression (3 & 4) & network heat maps*** show increased activity.

Adaptive Runtime Behavior for Word Count Running on 64 Nodes



1. I/O bandwidth
2. Key Throughput
3. Compression Throughput
4. Network Bandwidth
5. Reducer Throughput

Switch from ***network (3 & 4)*** to **I/O (1)** as the biggest performance bottleneck. File system contention results in a **6% performance degradation**

TeraSort Case Study

In-Depth Application Analysis for Weak Scaling

Non Adaptive Runtime Behavior for Tera Sort Running on 64 Nodes

1. I/O bandwidth
2. Key Throughput
3. Compression Throughput
4. Network Bandwidth
5. Reducer Throughput

The **network (3 & 4)** is underutilized until the **dataset ingestion (1)** is completed.

Adaptive Runtime Behavior for Tera Sort Running on 64 Nodes

1. I/O bandwidth
2. Key Throughput
3. Compression Throughput
4. Network Bandwidth
5. Reducer Throughput

The **file ingestion (1)** is faster and the **network (3 & 4)** is better utilized but it is more contended which leads to a slight performance degradation

U.S. DEPARTMENT OF ENERGY