Data-Intensive Computing on the M45 Cluster

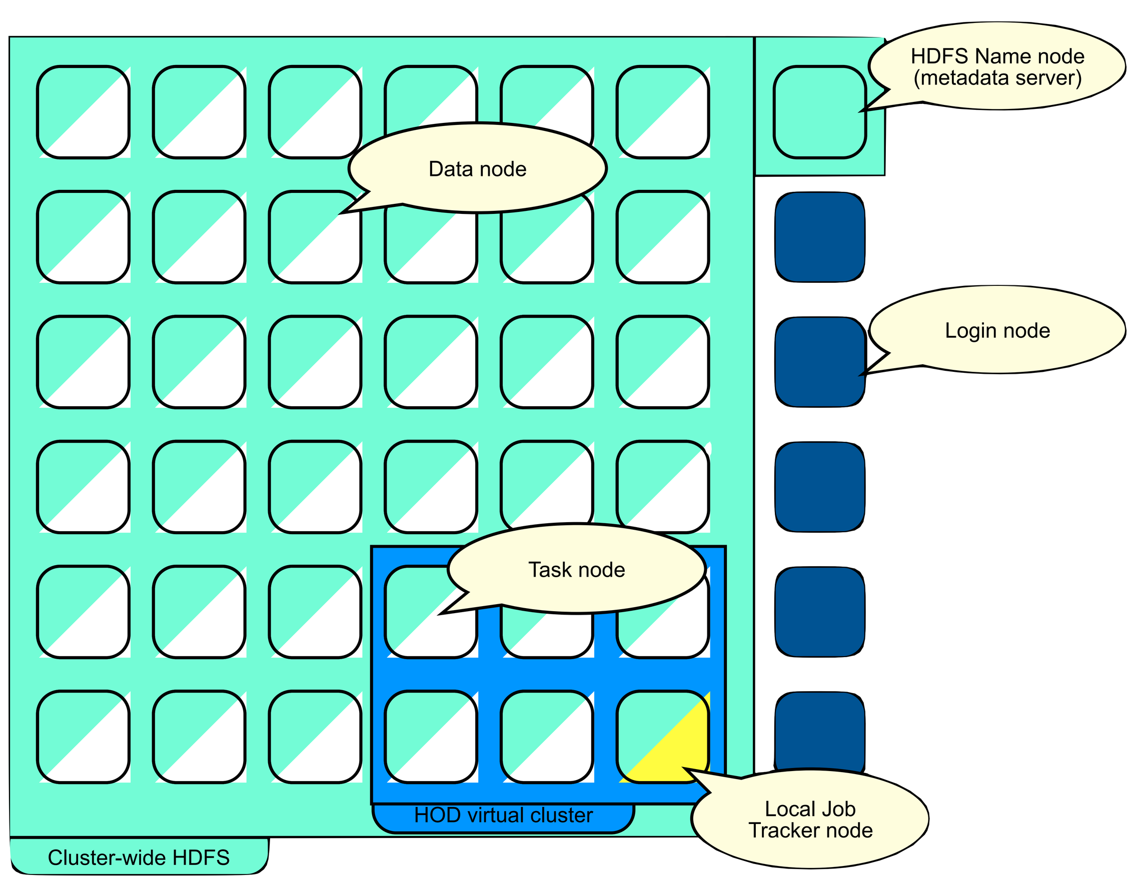
Julio López, Greg Ganger, Garth Gibson and Randy Bryant

Overview

- Exponential data growth: Multi-TB and PB size datasets
- Apps benefit from large data, e.g., Machine Learning
- Data analytics difficult at scale:
  - Hard to program, stress I/O and memory, frequent failures
- CMU/Yahoo! collaboration: M45 cluster (2400 cores)
- Rapidly enabled scaling up research problem sizes

M45 Cluster

- Total: 2400 cores, 1.8 TB RAM, 0.9 PB storage
- Cluster in a trailer
- 300 nodes: (8 cores, 6GB RAM, 4 x 750GB disks) / node
- GB Ethernet networking
- Hadoop stack: Linux, Hadoop M-R, HDFS. HOD, Pig
- Cluster-wide shared file system: HDFS



Applications

- Machine Learning / Data mining / Language Technology
  - American English web dataset (Callan)
    Select documents suitable for learning English
  - Grammar induction (Smith)
    Inferring language structures
  - Statistical Machine Translation (Vogel)
    Modern language translations, large training data
  - N-gram extraction (Mitchell)
    Creating corpora for language analysis
  - Understanding Wikipedia (Kraut)
    How Wikipedians collaborate?
  - Large-scale graph mining (Faloutsos)
    Analyzing graph structure of different web networks
  - Large-scale scene matching (Efros)
    Retrieve images from FLICKR & index
- Systems
  - Performance monitoring (Narasinham & Ganger)
    Automatic failure diagnosis
  - Parallel file systems for Hadoop (Gibson)
    Exploring other DFSs for Hadoop, e.g., PVFS, pNFS

Hadoop

- Implementation of Google's Map-Reduce (M-R)
- Open-source Apache project http://hadoop.apache.org
- HDFS: Hadoop Distributed File System
- Scalable associativity: Distributed "GROUP BY"
- Directly operatates on input dataset – no data loading
- Ability to process raw unstructured input
- Automatic input partitioning and task scheduling
- Scales data-bandwidth by moving computation to data
- Fine granularity failure handling

Experiences

- Good problem size scaling in a short period of time
- Learning curve:
  - Using the system, plugging things together
  - Parallelizing the apps. Mixing existing and new code
  - Loading data, preparing input, dealing with small files
  - Being good web crawlers
  - Debugging apps, tracking performance
  - Cluster sharing through user self policing
- Many ML apps lend well into the Map-Reduce model
  - Facilitates large-scale statistics computation
- Hadoop hides distributed programming complexity
  - Enables distributed and out-of-core processing
  - Good for unstructured, irregular and unordered input
  - Offers little benefit for ordered input (map-only tasks)
- Constrained computing model:
  - Map/Shuffle/Sort/Reduce or Map-only tasks
  - Coarse-grain lockstep operations (map/reduce waves)
  - Not natural for multi-dataset operations
- Good building block for distributed abstractions

Oportunities for Improvement

- File-system features
  - Allow post-creation writes (appends are now possible)
  - Multiple writers to non-overlaping offset-ranges
  - Using general parallel FS (Tantisiriroj et al.)
  - Dealing with many files (GIGA+, Patil et al.)
- Performance isolation (Wachs et al.)
- Performance monitoring "meta-analytics" (Tan et al.)
- Hadoop parameter configuration (Sambasivan et al.)
- Cluster management (Tashi project: CMU/Intel/Yahoo!)
  - Easier and efficient resource sharing and accounting
  - Storage / computation co-scheduling
- Hybrid storage/computation models

Carnegie Mellon

Parallel Data Laboratory