
MSST'08 Tutorial: Data-Intensive Scalable Computing for Science

What we learned with the M45 experience

Julio López

Parallel Data Lab -- Carnegie Mellon University

Yahoo! M45 cluster

- Companies have made resources available
 - Y! M45 cluster: 300 nodes, each 8 cores, 6GB RAM, 3TB
 - Intel Research Pittsburgh: 100 nodes, 8 cores
 - Google + IBM + NSF: shared facility
- Running Linux + Hadoop

M45 Application Projects

- Large-scale scene matching:
 - Retrieve images from FLICKR & index
- REAP project databases:
 - American English web database: Gather 100M web pages
 - Select documents suitable for learning English
 - Graded by difficulty and topic
- SMT in the cloud: Statistical Machine Translation
 - Modern language translations.
 - Needs lots of training data & many cycles
- Understanding Wikipedia:
 - How Wikipedians collaborate

M45 Support Projects

- Large-scale graph mining
Analyzing graph structure of different web networks
- N-gram extraction
Creating corpora for language analysis
- Grammar induction
Inferring language structures
- Performance monitoring
System infrastructure for performance data analysis and automated failure diagnosis
Using ML to performance prediction and system tuning
- Parallel file systems for Hadoop
Exploring other DFSs for Hadoop, e.g., PVFS, pNFS.

Science-related projects

- Earth Science related
 - Material ground model generation
 - Analysis of simulation-generated wavefields
 - Wavefield comparison
- LANL cosmic code comparison
 - Extracting Halos and Dark Matter properties
 - Comparing results from different simulation methods

M45: What we've learned

- There's a learning curve:
 - Programming: how to plug things together
 - How to mix existing legacy code & new
 - How to configure Hadoop
 - Being good web crawlers, experiential learning
- Dealing with the input: formats, small files, etc.
- **Achieved good problem size scaling**
... in a short period of time.

M-R & Hadoop strengths

- Simple, easy-to-understand programming model
- Good for unstructured data: customized parsing
- Powerful “GROUP BY” primitive
 - Unordered input
 - Suited for computing statistics, e.g., term frequency
- System - application separation
 - Distributed and out-of-core processing
 - Resilient failure handling
 - Enables co-location of storage and computation

M-R shortcomings

- Low-level primitive for distributed systems
 - A building block for higher-level abstractions
- Constraining pattern: M/S/R, M-only
 - What about recursive block transformations?
- Coarse-grained lockstep operations
 - No coordination between parallel tasks, no RPC
 - Streaming model? Service composition? queries?
- Little benefit for ordered data
- Not-so-natural multi-dataset operations
- Reading custom data formats

Hadoop development

- HDFS interface and semantics
- HDFS small file performance (container files)
- Cluster performance isolation, e.g., during shuffle
- Code maturity and performance issues:
 - Memory requirements
 - Collective operation coordination
 - Tangled feature implementation:
Programming model, scheduling, protocols, ...

Research opportunities

- **Systems aspect**
 - Storage/computation isolation (fault / sched)
 - Cluster management: Tashi project.
 - Automatic parameter configuration
- **Applications**
 - Using M-R model for data-analytics in science
 - FS and storage-computation models
 - Programming model for data-analytics in science