HadoopResearchProjects

Here are some research project ideas, engineering ideas for new participants, and areas where domain experts from other fields might add a lot of value by bringing their perspective into the Hadoop discussion.

- **Modeling of block placement and replication policies in HDFS**
  - Modeling of the expected time to data loss for a give HDFS cluster, given Hadoops replication policy and protocols.
  - Modeling of erasure codes and other approaches to replication that might have other space-performance-reliability tradeoffs.
- **HDFS Namespace Expansion**
  - Prototyping approaches to scaling the HDFS name space. Goals - Keep it simple; Preserve or increase meta-data operations / second; Very large numbers of files (billions to trillions) & blocks
- **Hadoop Security Design**
  - An end-to-end proposal for how to support authentication and client side data encryption/decryption, so that large data sets can be stored in a public HDFS and only jobs launched by authenticated users can map-reduce or browse the data. See HADOOP-xxx
- **Hod ports to various campus work queueing systems**
  - Hod currently supports Torque and has previously supported Condor. We would like to have ports to whichever system(s) are used on major campuses (SGE, ...).
- **Integration of Virtualization (such as Xen) with Hadoop tools**
  - How does one integrate sandboxing of arbitrary user code in C++ and other languages in a VM such as Xen with the Hadoop framework? How does this interact with SGE, Torque, Condor?
  - As each individual machine has more and more cores/cpus, it makes sense to partition each machine into multiple virtual machines. That gives us a number of benefits:
    - By assigning a virtual machine to a datanode, we effectively isolate the datanode from the load on the machine caused by other processes, making the datanode more responsive/reliable.
    - With multiple virtual machines on each machine, we can lower the granularity of hod scheduling units, making it possible to schedule multiple tasktrackers on the same machine, improving the overall utilization of the whole clusters.
    - With virtualization, we can easily snapshot a virtual cluster before releasing it, making it possible to re-activate the same cluster in the future and start to work from the snapshot.
- **Provisioning of long running Services via HOD**
  - Work on a computation model for services on the grid. The model would include:
    - Various tools for defining clients and servers of the service, and at the least a C++ and Java instantiation of the abstractions
    - Logical definitions of how to partition work onto a set of servers, i.e. a generalized shard implementation
    - A few useful abstractions like locks (exclusive and RW, fairness), leader election, transactions,
    - Various communication models for groups of servers belonging to a service, such as broadcast, unicast, etc.
    - Tools for assuring QoS, reliability, managing pools of servers for a service with spares, etc.
    - Integration with HDFS for persistence, as well as access to local filesystems
    - Integration with ZooKeeper so that applications can use the namespace
- **A Hadoop compatible framework for discovering network topology and identifying and diagnosing hardware that is not functioning correctly**
- **An improved framework for debugging and performance optimizing hadoop and streaming Hadoop jobs**
  - Some suggestions:
    - A distributed profiler for measuring distributed map-reduce applications. This would be real helpful for grid users. It should be able to provide standard profiler features , e.g. number of times a method is executed, time of execution, number of times a method caused some kind of failures, etc; maybe accumulated over all instances of tasks that comprised that application.
- **Map reduce performance enhancements**
  - How can we improve the performance of the standard Hadoop performance sort benchmarks?
- **Sort and shuffle optimization in MR framework**
  - Some example directions:
    - Memory-based shuffling in MR framework
    - Combining the results of several maps on rack or node before the shuffle. This can reduce seek work and intermediate storage.
- **Work load characterization from various Hadoop sites**
  - A framework for capturing workload statistics and replaying workload simulations to allow the assessment of framework improvements.
- **Other ideas on how to improve the frameworks performance or stability**
- **Benchmark suite for Data Intensive Supercomputing**
  - Scientific computation research and software has benefited tremendously due to availability of benchmark suites such as NAS Parallel Benchmarks. This was a kernel of 7 applications, starting with EP (embarrassingly parallel) to SP, BT, LU (reflecting varying degree of parallelism and communication patterns). A suite for data-intensive supercomputing application benchmarks would present a target that Hadoop (and other map-reduce implementations) should be optimized for.
- **Performance evaluation of existing Locality Sensitive Hashing schemes**
  - Research on new hashing schemes for filesystem namespace partitioning: http://en.wikipedia.org/wiki/Locality_sensitive_hashing
- **An alternate view of files a collection of blocks**
  - Propose an API and sample use cases for a file as a repository of blocks where a user can add and delete blocks to arbitrary parts of a file. This would allow holes in files and moving blocks from one file to another. How does this reconcile with the sequence-of-bytes view of file? Such an approach may encourage new styles of applications.
  - To push a bit more in a research direction: UNIX file systems are managed as a sequence-of-bytes but usually (and in Hadoop's case exclusively) used as a sequence of records. If the filesystem participates in the record management (like mainframes do for example) you can get same nice semantic and performance improvements.