# *Big Data Paper Summary:*

*The Google File System (by: Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung)*
**AND**
*A Comparison of Approaches to Large-Scale Data Analysis*

---

GREGORY ABBENE

PROFESSOR LABOUSEUR

CMPT-308 SPRING 2014

DUE: MAY 2ND 2014

# *Main Idea:*

- **Google File System (GFS) is a "scalable distribution file system for large distributed data-intensive applications."**
- **In other words, it is a Big Data platform/system with the goal of optimization of aggregate performance at a cost efficient hardware and software level.**
- **GFS is able to cluster hundreds of terabytes of storage across thousands of disks and machines that are accessible by hundreds of clients.**
- **GFS utilizes an agile architecture to help report metrics for "micro-benchmarks" and real world uses (providing metadata), while attempting to optimize the balance between performance, scalability, reliability, and availability.**

# *Implementation*:

- **GFS has overall architecture to store all metadata (similar to PostgreSQL's System Catalog, but on a much larger scale) into one master that handles clustered data from chunked locations in its memory.**
- **In addition to the single master, multiple "chunkservers" with a globally unique "64 bit chunk handle" stores local fixed-size files on local disks, holding up to 64 MB of block size for the file system to map to memory—this is much larger than the standard 4-8 KB.**
  - **These "chunk locations" are hosted on cost-efficient hardware with a Linux-based OS, and they are monitored by "HeartBeat" messages to communicate with the host and maintain data consistency.**
  - **Through "checksumming" the chunkservers can detect corrupt data storage to assure data integrity, based on a precise replications of crucial data at the disk or IDE subsystem level.**
  - **The use of "lazy space allocation" utilizes internal fragmentation to assure no space is wasted within these relatively large chunk sizes.**
- **The master hosts a vast logging mechanic, that consistently tracks all files and chunks. Routine checkpoints are set up to maintain the size of the log that compact the given B-Tree directly into memory.**

# *Analysis*:

- **I think that the GFS uses a very scalable system, starting with the use of their master system to monitor the sometimes thousands of disks and machines.**
- **The cost efficient use of hardware systems imbedded with Linux is a valuable asset to this system, as it provides more flexible real-world use cases. Many of these big data systems are built for large enterprise systems, however I think that GFS makes the use of Big Data more accessible and realistic to use.**
  - **Even though this isn't particularly sold, as it is used as in-house for Google, it is a valuable model for other systems.**
- **GFS seems to have successful autonomic computing principles, meaning it's ability to monitor and self-diagnose problems, and this is a massive appeal.**

# Comparison with *A Comparison of Approaches to Large-Scale Data Analysis:*

- **The concept of "cluster computing" is exactly correspondent to GFS…as large number of "low-end servers" are utilized. This can be congruent to the use of "cheap" hardware with Linux OS that are used in GFS.**
- **The concept of partitioning local data into a shared system, is similar to GFSs clusterservers that replicate and parallel onto the main server.**
- **That being said instead of a File system, the use of parallel relational DBMSs (such as Microsoft SQL Server) are much more advanced then architecture of GFS.**
- **Both systems also boast fault tolerance and benchmarking capabilities that help produce meaningful metadata.**

# Advantages and Disadvantages:

***ADVANTAGES***:

- **Similar utilization of cluster computing for cost efficient scalability**
- **Metadata is stored and accessible through a single master, as opposed to parallel DBMS's MR model**
- **Utilizes B-tree indexing and "Heartbeat" messaging for storage efficient log filing, replication decisions, and data integrity checks.**
- **The single master system provides for a strict monitoring of chuckservers through checksums and metadata**

***DISADVANTAGES***:

- **The parallel SQL DBMS has the ability to run has querying capabilities**
- **Not using complex data architecture and models that make more useful for programming.**
- **Lack of overall constraints of data distribution, that limits network connections of the nodes to the given cluster**
- **Lack of flexibility of data use that is used by the parallel DBMS.**