Managing High-Bandwidth Real-Time Data Storage

David Bigelow, Scott Brandt, John Bent, HB Chen

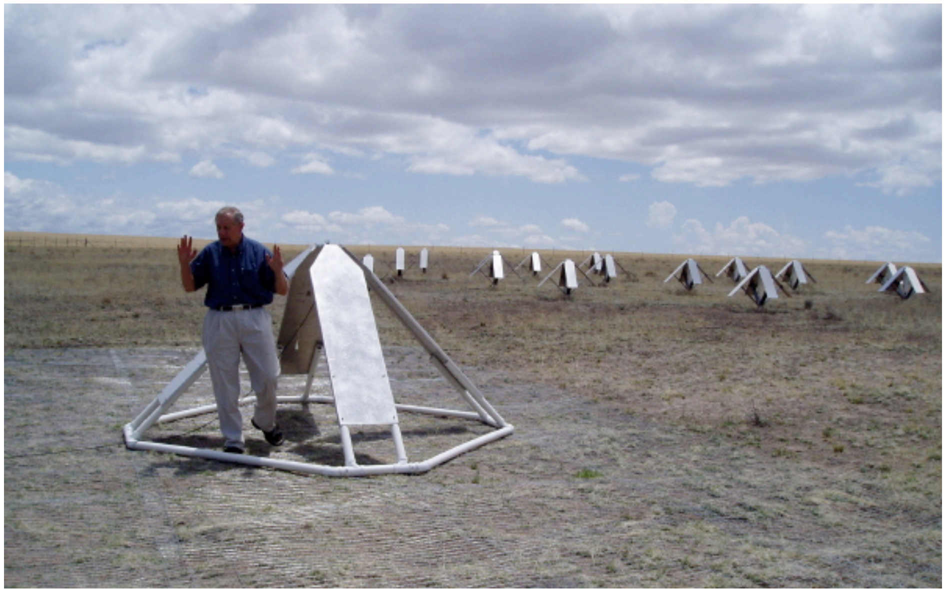
Data Capture at High Speeds

- **Problem: Temporary storage of “lots” of data**
  - Example: Astronomical observations
  - Example: Network traffic capture
  - Trivial Example: TiVo
- **Most data is worthless over the long run**
- **There’s too much of it to go into permanent storage**
- **But sometimes the data is actually worthwhile**
  - …and so were the last ten minutes of it, but you didn’t know that until just now
- **Need a system that can address these problems**

Motivating Project: Long Wavelength Array

- **Low Frequency Radio Telescope**
- **Geographically distributed but synchronized**
- **Most collected data is just noise**

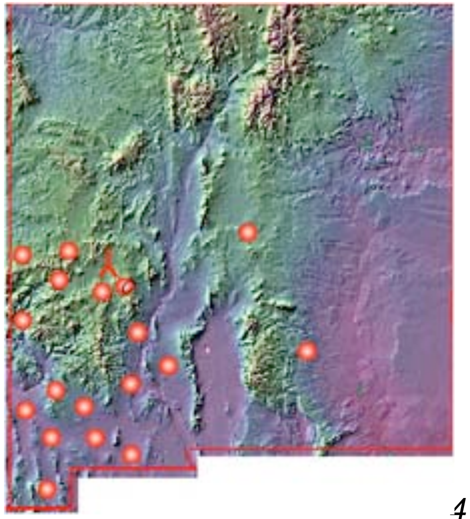
- **Basic Statistics:**
  - 53 stations (initially)
  - 400 km base line
  - 580 Mbit/sec data rate
  - ~30 Gbit/sec total



Requirements

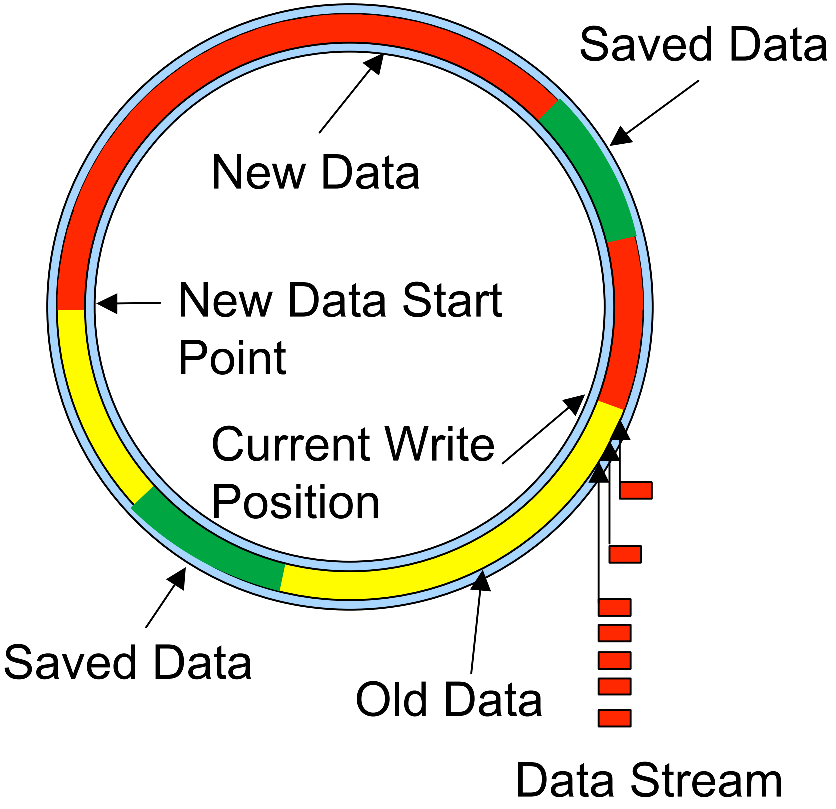
- **Quality of Service Guarantees**
  - Incoming data *must* be recorded on the first (and only) transmission at a set bandwidth
  - There needs to be a mechanism to read data back off as well
- **Reliability**
  - Data cannot be regenerated and thus must not be lost
  - QoS must be maintained in the face of hardware failure
- **Infrastructure**
  - Efficient use of commodity hardware
  - Must be able to run in a desert shack
  - Scale to hundreds or thousands of units

Right: Locations of LWA stations over southwestern New Mexico



Our Solution: Ring Buffer

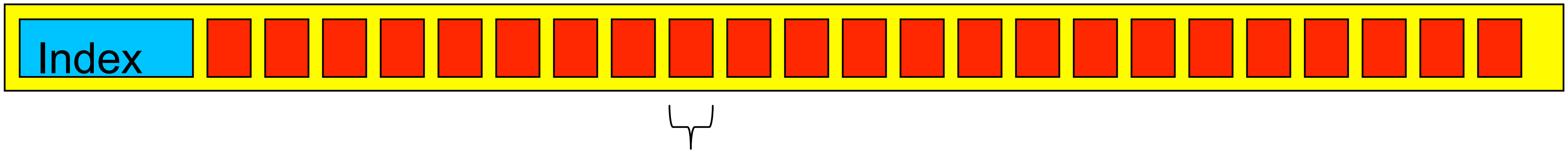
- **Fixed Size**
  - Allows "X" time units of storage
  - Very little bookkeeping required
- **Limited Lifetime**
  - Data is quickly overwritten if not specifically preserved
  - No "cleaning up" needed
- **Limited Indexing/Metadata**
  - Only a small amount of primary indexing is needed, and traditional metadata is barely needed at all



A New Filesystem

- **Many standard filesystem features useless**
  - No need for file creation, deletion, stat, etc.
  - Only ever one writer (though there may be several readers)
  - Most metadata is useless
  - Indexing is vastly simplified
- **All operations done on large blocks**
  - Aggregated writes for maximal I/O performance
  - Fragmentation problems minimized
- **File system never “shuts down”**
  - No need to maintain an on-disk index
  - Disk head movement at a minimum
  - Can reconstruct index again at startup, but time is not critical

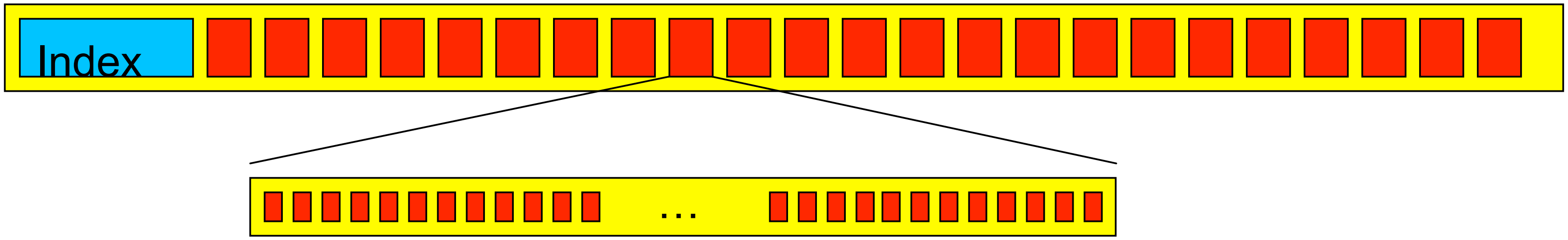
Big Data



One data chunk

- **Basic indexing: one data chunk, one ID**
  - Easily maintained in main memory with big enough chunk size
- **Fixed size: never need to think about “sub-chunks”**
  - Always read and write on fixed-sized chunks of data
- **Simple parameterization**
  - Configuring such a setup requires only the chunk size and ID information

Small Data



One data chunk has lots of individual pieces of information

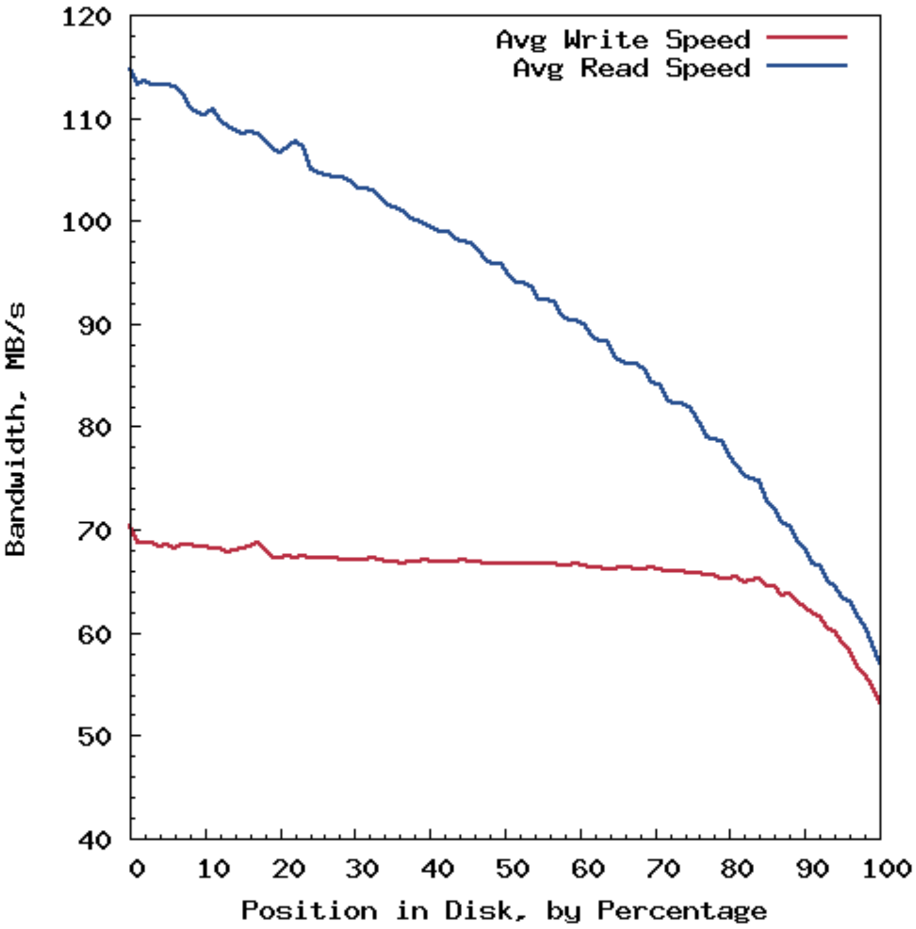
- **Full index cannot be kept in main memory**
  - Need to store secondary indexing information on disk
- **Variable size**
  - Minor internal fragmentation
  - Might want smaller portions of data read or preserved
- **Complex parameterization**
  - Multiple things to index on

Prototype and Testing

- **Prototype System: Mahanaxar**
  - Currently runs on single hard drives for both big and small data
- **Primary comparison: flat file system (ext2)**
  - Initial testing on several different filesystems
  - ext2 has slightly better performance
- **Database comparisons show very poor performance**
  - As the system ages at 99.9%+ capacity, database speed collapses
- **Performance testing over several hard drives yielded similar data**
  - For these results, one particular hard drive is used for all comparisons (a 1.5 TB Western Digital SATA drive)
  - All results are from a system fully-populated (99.9%+) with data

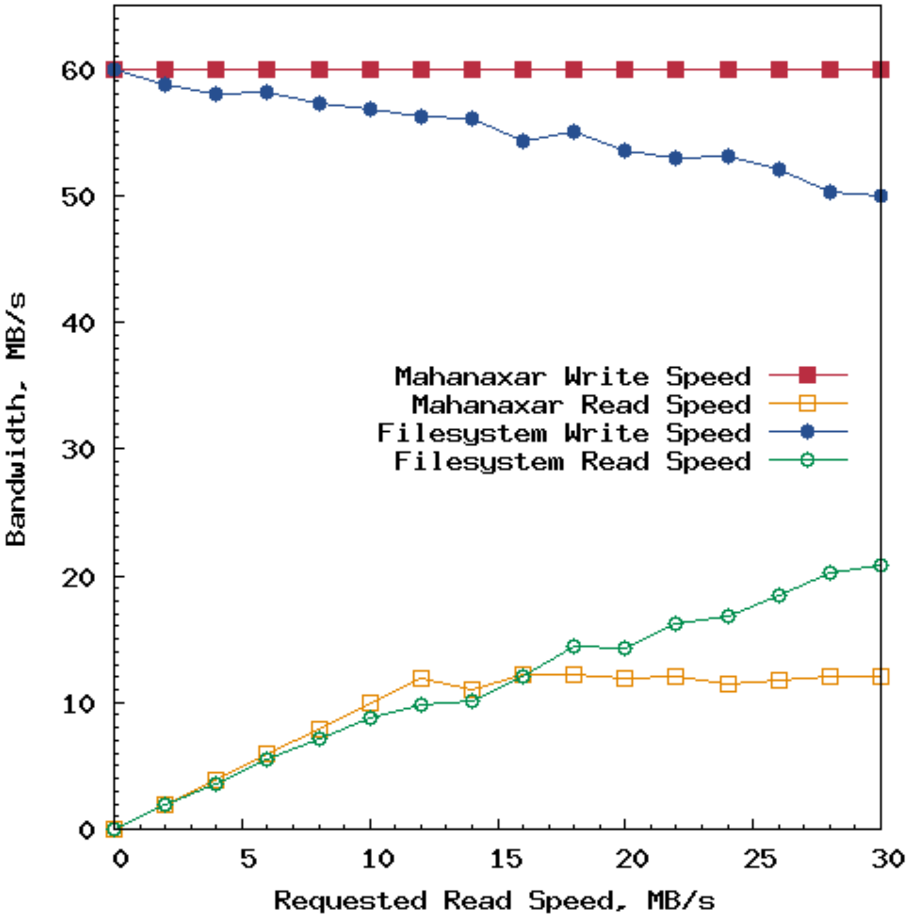
Disk Profiling

- Performance degrades over course of disk
- There is a sharper performance degradation towards the end of the disk
- May only want to use portions of the disk to maintain higher overall performance



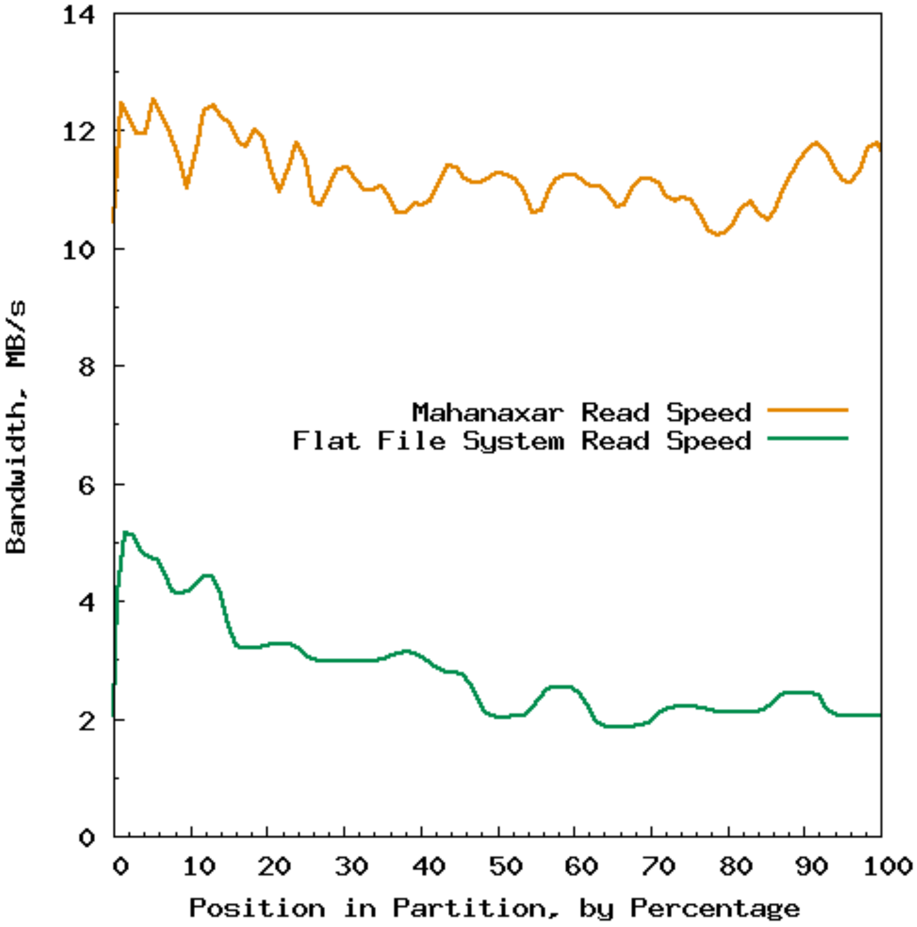
Mahanaxar vs. Flat Files

- Requested write speed: 60 MB/s
- Ordinary filesystems mechanisms used for access in filesystem testing
- Maximum theoretical read bandwidth available is ~11 MB/s



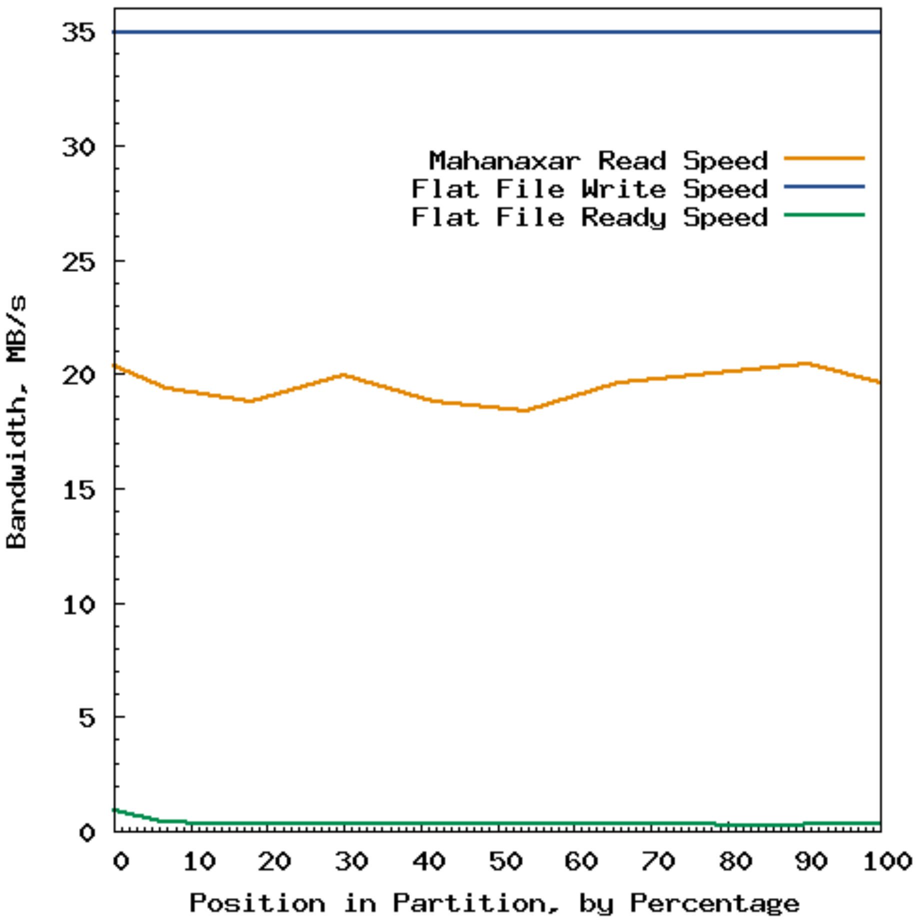
Mahanaxar vs. Flat Files with Constrained Access, 60 MB Elements

- Both systems maintain 60 MB/s requested write speed (not shown)
- Mahanaxar has 3-4 times as much spare bandwidth for reading
- Large element size provides best possible circumstances for flat file system



Mahanaxar vs. Flat Files with Constrained Access, 1 MB Elements

- **Requested write speed still 60 MB/s**
- **Mahanaxar maintains 60 MB/s (not shown)**
- **Flat files only manage about 35 MB/s**
  - Nearly half of the data is dropped
- **Flat file system available read bandwidth is minimal**



Questions

- What happens when you run a commodity hard drive 24/7/365 at 99.9%+ capacity?
- How would one control ten thousand nodes simultaneously?
- Other Questions?