desiderata

• operate scalably
  – petabytes of data
  – larger than RAM, disk i/o required
• operate economically
  – minimize $ per cycle, ram, & i/o
  – thus use network of commodity PCs
• operate reliably
problem: seeks are expensive

- CPU & transfer speed, RAM & disk size
  - double every 18-24 months
- seek time nearly constant (~5%/year)
- time to read entire drive is growing

moral:
- scalable computing must go at transfer rate
two database paradigms: seek versus transfer

- B-Tree (Relational Dbs)
  - operate at seek rate – \( \log(N) \) seeks/access
- sort/merge flat files (Lucene, MapReduce)
  - operate at transfer rate – \( \log(N) \) transfers/sort

- caveats:
  - sort & merge is batch based
    - although possible to work around
  - other paradigms (memory, streaming, etc.)
example:
updating a terabyte DB

• given:
  – 10MB/s transfer
  – 10msseek
  – 100B/entry (10B entries)
  – 10kB/page (1B pages)

• updating 1% of entries (100M) takes:
  – 1000 days with random B-Tree updates
  – 100 days with batched B-Tree updates
  – 1 day with sort & merge
problem: scaling reliably is hard

• need to process 100TB datasets
• on 1 node:
  – scanning @ 50MB/s = 23 days
  – MTBF = 3 years
• on 1000 node cluster:
  – scanning @ 50MB/s = 33 min
  – MTBF = 1 day
• need framework for distribution
  – efficient, reliable, easy to use
MapReduce: sort/merge based distributed computing

- best for batch-oriented, offline
- naturally supports ad-hoc queries
- sort/merge is primitive
  - operates at transfer rate
- simple programming metaphor:
  - input | map | shuffle | reduce > output
  - cat * | grep | sort | uniq -c > file

- distribution & reliability
  - handled by framework
comparison of current scalable database strategies

- **partitioned RDBMS**
  - access: +online
  - distribution: -custom
  - partitioning: -static
  - updates: -slower
  - schema: -static
  - joins: -slow/hard

- **MapReduce**
  - access: -offline
  - distribution: +native
  - partitioning: +dynamic
  - updates: +fastest
  - schema: +dynamic
  - joins: +fast/easy

- **HBase/BigTable**
  - access: +online
  - distribution: +native
  - partitioning: +dynamic
  - updates: +faster
  - schema: -static
  - joins: -slow/hard
Hadoop

- Apache project
- includes:
  - HDFS – a distributed filesystem
  - MapReduce – offline computing engine
  - HBase (pre-alpha) – online data access
- Y! is biggest contributor
- still pre-1.0 release
  - but already used by many
over to Eric...