Databases

Patrick Nolan (Stanford)

SSC-LAT Science Tools Workshop
June 12-14, 2002, SLAC

Databases

- At Stanford we have focused on conventional RDBMS systems that use SQL. There are other things out there (object-oriented, in-memory, etc.) which might be worth a look.
- Free databases: Postgres and MySQL are two well-supported freeware products with large, active user bases. MySQL is a simple, basic product, easy to use but short on features. Postgres has object-relational features that could be useful.
- Commercial databases: Stanford/SLAC has a site license for Oracle, so we can use it with no visible cost to us. We hope that this arrangement will continue. What's the situation at other institutions? Nobody ever got fired for requiring Oracle, except state employees. We have also obtained an evaluation copy of IBM's DB2, but the purchase price is probably out of our range, so we haven't done anything with it.
- What Oracle gives you: Here are some of the features that are found in Oracle, but not the free databases:
  - Tablespaces allow the database to be split into sections that are stored separately.
  - Complete control over placement of data files. Multiple files can be used for a tablespace. File sizes can be specified.
  - Index, temporary, and rollback segments can be stored separately if desired.
  - Large tables can be partitioned for separate storage and efficient indexing.
  - Oracle Net allows databases on separate machines to be shared in a nearly transparent way.
  - Hundreds of decisions need to be made about data layout and parameter settings. People make their entire careers doing this.
- Object-relational advantages: Relational databases have an inflexible data model, but Object Oriented databases are struggling with performance issues. Object-relational is a buzzword which covers
  - User-definable data types, including "struct"-type objects.
  - User-definable functions written in C or a proprietary language.
  - Extendable indexing methods.
- Indexing issues: When looking up data, there can be huge differences in performance, depending on how it's ordered and on the use of an index. Every DBMS has a complex query optimizer. Much of the tuning process involves second-guessing the optimizer, perhaps convincing it to use an index when it might choose otherwise. There are various kinds of indices: hash tables, B-trees, R-trees (for two-dimensional data), etc.
- Ingestion: There are various ways to get data into a DBMS. Simple reading of a binary file is not one of them. ASCII is the easiest form to handle. Oracle doesn't use the IEEE floating point format, so binary files are not optimal at all. We have found that all systems can ingest a day of fake data in less than a minute if it's done intelligently, using special tools which shut off transactions and rollbacks.