LIST UPDATE PROCESSING (LUP) - SOLVING THE SEQUENCE DATABASE UPDATE PROBLEM

R.DOELZ, F.EGGENBERGER
BioComputing Basel, Biozentrum der Universität, Klingelbergstrasse 70
4056 Basel,Switzerland

Sequence databases of today require frequent updating. Mirror procedures to copy incrementally updated databases as cumulative sets are the preferred method and can be implemented by straightforward scripting. However, limited bandwidth of networks and the increase of data require more powerful paradigms to reduce the workload reliably. We suggest the List Update Processing (LUP) principle. The system has been implemented on an experimental basis to update the Swiss EMBnet Node (BioComputing Basel, CH) with data from the European Bioinformatics Institute (EMBL Outstation, Hinxton Hall, UK). The results obtained from the prototype suggest to expand the system to several sites.

1 Introduction

Sequence databases grow with incredible speed. In order to maintain a database locally, it is required to transfer the database or parts thereof from a remote provider to the local system, format it accordingly, and utilize the growing information within application programs. Already some years ago, trends of growth indicated that the emerging CD-ROM technology will face limits in a few years, as the current density of data on a single disk is limited to about 680 Mbytes of data. Additionally, production and shipping of databases on CD-ROM can hardly be accomplished in less than two weeks, which would result in a considerable backlog and significant cost. One of the prime goals of the European Molecular Biology (EMBnet) project [1] is to accomplish database updating via electronic networks.[a]

Three major problems are to be tackled in order to achieve seamless data processing:
1. Data selection at the provider side: Today's sequence databases are prepared by exporting new data entries from a database management system.
2. Transfer of data from provider to customer
3. Integration and formatting of the data at the customer side.

In this paper, we describe a novel method to synchronize databases in an improved set of client/server processing routines.

[a] More information is available at http://www.ch.embnet.org/embnet.news/

2 Known methods

Nursing[2]

Early procedures relied on the transfer of blocks of data, prepared from single entries, which were pushed to the customer. The selection was accomplished by creation date of the individual export files. Originally designed for data transfer via X.25 technology, the transfer of the data was accomplished with the FTP program in later stages. For this purpose, the customer had to allow the provider to approach a server accessible on non-anonymous basis (username/password authentication).

The integration and formatting process has not specifically been solved but was performed via the file system . Addition and replacement were achieved this way, as individual entries created one file each.

NDT[3]

The Network Data Transfer (NDT) protocol, based on the push paradigm, initiates the transfer on the provider side after proper selection of entries. Procedures are is similar to the 'nursing' approach. The provider uses a special-purpose client to talk to a special-purpose server, and uses IP number and protocol-inherent authentication.

FTP

The use of the ftp protocol has become very popular with the growing bandwidth of international networks, which allowed that the cumulative set of data could be transferred in one run. Besides the waste of bandwidth caused by this procedure, both disadvantages of insufficient success control and flexibility are maintained. However, as the customer approaches the provider to maintain service, the principle followed is a poll paradigm. The database providers do also offer to download incremental updates, which are prepared batchwise in continuous fashion on daily or repository. There is no control on success or completion of the transaction, and deletion cannot be processed either. As the communication is initiated by the customer (client), anonymous ftp (no authentication) is possible.

The data integration of an FTP stream is left to the client - currently, no procedures are provided to manage the data. Data reformatting of cumulative updates on script-level basis is the most frequently used option.

3 Principle of the List Update Processing

The following assumes that the goal of the update procedure is to create a perfect copy of a master database. Sequence databases of various sites can theoretically

Table 1: Sets of sequence database updates

Site	Name of file	Sets	weekly/other incremental
EBI	cumulative.dat	non-sorted	yes
NCBI	gbcu.flat	sorted	yes
Basel	emnew.dat	xembl (new EBI data), xxembl (updated set), gb_new (exclusion set)	no
NDT site	em_new.*	(one per entry) single files	n.a.

judged for quality by the number of sequences contained in this database. This figure is, however, not necessarily sufficient to characterize the data set exhaustively, as the temporarily changing nature of the set is neglected. The following outlines the requirements to handle sufficiently abstracted information which shall be utilized in order to duplicate databases over networks.

3.1 Lists

As mentioned, the number of entries in a database can only provide a rough estimate whether the mirroring performs well. However, as both duplicate or outdated entries would contribute to the total number, it is required to list the names of the entries also. Comparing lists of entries, however, does still allow that some entries are 'outdated', which means that the update of a given entry has not been propagated as required. Therefore, the following data are required to characterize a data set sufficiently well:

- number of entries
- names of the entries
- a property of the entry which characterizes the status of this entry (e.g., version number or date stamp).

In order to process updates on a subentry level (atomic updates), it were advantageous to add a checksum which characterizes the data set by contents rather than by number or property. Due to the overhead caused by this procedure, we prefer to externalize the atomisation of data (see Discussion).

3.2 Properties of updates

Data sets available today are usually available as sequential files alphabetically sorted by their master key such as the entry name. However, each site will possibly have more than one single set, and unfortunately the criteria for a given set are different. Table 1 lists the current composition of three sets in exemplary fashion.

The integration of incrementally received updates requires the deletion of entries which are already in the current set in a previous version , and the insertion of the new entry. Sequentially oriented files, however, do not allow entry substitution as the volume occupied by the entry is variable. We have created the DBTOOLS package earlier [4] which used a indexing schema similar to SRS [5] in order to allow fast access to sequential data. The overhead caused by this mechanism is considerable and it is difficult to justify just another indexing schema. In addition, before the transaction can take place, a backup of the original set is always required.

The **List Update Processing (LUP)** approach is different and solves the problem elegantly. Instead of requiring a backup and working on the original file, the LUP tools create a new file from the live data, which is validated before it is used to replace the production set. LUP tools reduce the update manipulation to lists rather than full data sets. Lines in these lists are sufficiently informative to allow entry characterization by name, accession number, date and version. Instead of updating and mirroring data sets , the goal is to mirror the lists of these data sets are mirrored. The effect of such a list updating is that the unit of change is a defined entity (a single line), which can be handled much more easily than full data sets. Sorting and merging of lists is readily implemented by standard tools of most operating systems. In order to validate that a given mirror of the database is as complete as possible, it is sufficient to prove that the lists created from the databases are identical.

3.3 Communication

As the ftp protocol or other known tools do not allow that individual data sets are created on the fly, the customer must have a tool to create the required data on demand.

The Hierarchical Access System for Sequence Libraries in Europe (HASSLE) [6] has been tailored to meet this task. Security, access control and data compression are provided as features which facilitate the transfer. In combination with the LUP tools, we have therefore employed a HASSLE provider schema which allowed to create a fault-tolerant system of servers which will get the potential customer the best reply possible. In order to avoid unnecessary data duplication, we choose the HASSLE security feature of individual keys bound to site addresses [7].

3.4 Implementation of LUP routines

Based on a core routine library, about 30 individual programs are available as LUP toolset. Specific database formats are EMBL and GENBANK formats on the provider and customer side. On the customer side, the NBRF/GCG program format is also supported, together with a 'breaking' utility to split long sequences into the currently supported limited length of sequences.

In particular, we have implemented the following functionality:

- Listing of flat/formatted files
- Comparisons of listing files
- Merging of listing files
- Subtraction, extraction and specific manipulation of lists
- Resorting of lists according to lists
- Counting of sequence, reference and flat files
- Composition of new flat files based on listing files
- Composition (merging) of flat or formatted files based on mixed listing files

3.5 HASSLE

HASSLE has been described earlier [6,7] and and is readily available [b] to run the required programs as provider in the established provider environment. Based on the available raw data material, we have implemented a new service on experimental basis: *SEND_EBI*, for sending data updates in a single file ('cumulative.dat'). It should be noted that the provider does only send unmodified generic data. The transmission of formatted data is not anticipated in order to avoid data loss or falsification.

Two different actions are achieved in the services mentioned above depending on the phase of the process: the **listing of the provider data**, and the

[b] our FTP server is at ftp://ftp.switch.ch/mirror/embnet-ch/bioftp-sw/hassle/

extraction based on a 'to-do' list. The 'to-do' list processed by the HASSLE provider has to be submitted accordingly by the customer. If resource or bandwidth usage is an issue, the size of the data returned in a single HASSLE run might be restricted. Such a measure usually reflects a problem, but as the LUP principle inherits synchronization, the processing of partial batches will not affect the quality of the data.

3.6 Reliability

We used LUP in the past year to update our copy of the EMBL database. It was found that the synchronization with entire lists (as opposed to list updates only) was required on periodical basis, in order to

- enforce synchronization
- allow for deletions
- compensate malicious operational effects

The latter category (commonly referred to as 'bugs') is possibly irrelevant in an ideal world. However, we observed a wide variety of problems in incremental updating. FTP-based methods which we apply simultaneously to GENBANK updates showed such errors in rare occasions, however, remain undetected due to the lack of recovering mechanisms.
The LUP-based system had a time delay of two days at most and did not show any operational flaws . However, logistic problems (such as rejected transfers due to the fact that files at the provider side were busy due to data processing) seem to be more limiting to the set-up than failures of data transfer itself. On a 200 day average, 92of all days had the perfect mirror of the database, 3side (as deletions were processed weekly rather than daily) and most of other days suffered from 'operational' problems such as physical power outage, disk problems and similar effects which were compensated on a subsequent day. This 'insufficiency' rate of 5measured at the application side (i.e., in the application package format) and therefore look less advantageous than the data for the raw data mirror. Data on the application side were never affected in the sense of erroneous sequences but rather did just not do the update.
The time delay caused by 'pausing' of updates and large bulk transfers proved to be feasible as the option to segment list updates in chunks can be added on demand. Due to the reliability we observed even in the megabyte range, we dropped this option but are prepared to reinvoke it in case of need.

3.7 Adaptation to other systems

The LUP principle relies on various key elements:

- local manipulation
- protocol transfer
- remote manipulation

Each of these three elements can be utilized separately, or can even be replaced by already existing technology. The flow schema as described below refers to our reference implementation of all three elements.

4 Details of the LUP implementation: Flow schema

The updating of sequence databases via LUP tools requires a suitable set-up at both the provider and the customer side.

4.1 Customer side

The basic operational principle is to allow the customer to ask the provider for the creation of a list update. Once this file is received and properly processed, the customer will compare his own, current entry list with the required entry list and subsequently create a list of missing data ('to-do' list). This list is then forwarded to the provider with the request to send the complete data listed therein. The customer will create a merged list from own and newly obtained list data. Comparison with the already existing own data and the merged list results in the request to transfer the missing data.

4.2 Provider side

The basic operational principle is to allow the provider to return a list update or an update data set on request, respectively.

4.3 Propagation of deletion

The propagation of updates is currently either an addition or a modification transaction. A deletion can theoretically be described as a mutation of an entry to zero size. However, a deletion of an entry is not necessarily experienced as a transaction on the provider side. Instead, manual deletions might cause a file to be missing, or a missing file will no longer be part of the entire cumulative update file. Therefore, to process deletions exclusively on the basis of lists , it is required to transfer the entire current entry list rather than an update only on a periodic basis. A much more advanced and elegant procedure were possible if the provider could prepare a list of to-be-deleted entries which then can be processed in a dedicated LUP tool. Future collaboration with institutes like EBI will possibly allow for such a procedure.

5 Discussion

The update of databases via networks faces limits if the file to be copied exceeds the size of a dataset which can be transferred without experiencing problems in the data transmission. Time-outs on lines with low bandwidth were observed frequently in previous years. Today, the sheer size of the sets (even in compressed form, 100 Mbytes or more) appears to be the major problem. The limiting factor is the disk space at the customer site: Receiving an update requires two to three times the space of the production database.

5.1 Scope and limitations of incremental updates

As mentioned, updates to sequence databases can be classified into three types (new data, updates to new data and changes to already sent data). Rescoring these options, we get

- New data (to be added)
- Modifications of data (to be merged)
- Deletions of data (to be removed)

As only the second kind of data does actually change data sent earlier, the 'updates' as understood by biological researchers in the classical sense will be most probably new additions (first type). Classification as 'updates' to data of another data set (such as the full release of a sequence database) will need to be handled by the corresponding application. At our site, we split the 'new data' into really new data and those which have been updated with respect to the established release. A sequence database update section, therefore, can most beneficially be processed as only those sequences which are really new are used, and the inclusion of updates to the existing data would show as duplicates in the searching result.
For specific applications, we have experimentally set up a system operating with LUP tools. This approach performs the transactions on two different sets from one package of updates. We have confirmed that the data can be remerged at any time to maintain compatibility with other connections.

5.2 Integration

It should be emphasized that the selection which entries shall be updated, as well as the transmission of these entries (as currently handled by NDT, FTP or any other method) do not tackle the most important issue of the integration of these entries into the production database. LUP tools allow the integration in seamless fashion. As the processing of lists is intrinsically used in the merging of the database, the integration of data is no longer an isolated matter and can be dealt with as part of the updating process rather than an isolated item.

5.3 Flexibility of the LUP approach

The List Update Principle is currently implemented on the basis of very simple toolset. The basic idea of the implementation is to demonstrate the benefit of list processing. We do hope that, once the method as such gets more widely accepted, more advanced tools will become available to facilitate the creation of lists. Whereas the processing of the list is fairly straightforward, the creation of lists proved to be a sophisticated issue if non-sorted data files are encountered. As we process data in alphabetic fashion at our local site, the cumulative file as maintained at EBI showed to be alphabetically only in part, as it is assembled from three separate, sorted files. Attempts to extract this file with an alphabetically sorted list with one of the LUP programs showed to become an unaffordable input/output intensive process once the extraction list grew beyond a few hundred entries. Therefore, we had to create a tool which resorts a list of entries according to another list.

The previous release of our DBTOOLS package[4] made use of special-purpose index files. Besides the overhead of index creation, the code for generating a special retrieval schema is fairly complex and possibly requires more resources for maintenance than can be spent. The LUP tools, therefore, rely on very simple, serial processing implementations of code which is maintainable also by inexperienced programmers.

The retrieval of sequences from an unsorted database based on a sorted list will also be possible by state-of-the-art retrieval systems like SRS[5] once the data are sufficiently indexed. Further development on this is in progress. However, the effort to maintain indices for data maintenance (rather than data retrieval for customers) still requires some careful redesign considerations of the specific set-up.

Tool development at the database providers will allow promising development of the LUP principle. E.g., a relational database system will know best which entries are needed. the straightforward software in our reference implementation might benefit from these lists, and eventually make the provider handling more efficient. As the customer, usually, does not use relational systems, non-specific tools will always be needed.

5.4 POLL paradigm vs. PUSH paradigms

If the PUSH paradigm is to be followed the customer as the primarily interested partner will have to provide facilities that the provider can send the data on a periodical or semi-periodical basis. Depending on the protocol employed, the provider will encounter the following feedback (sorted by increasing complexity):

- Success status of a transmission
- Maintenance of a date of the last successful transmission
- Maintenance of a transaction number or name of the last successfully transmitted entry
- Feedback of successfully processed transactions
- Feedback of successfully application-ready formatted database

The more complex the feedback shall be, the more information has to be computed at the customer side and must be accordingly transmitted to the provider. Depending on the kind of these computations, these require that the provider is capable to execute or at least trigger these actions. This raises an important security problem: Both data security (possible corruption of data) and operational security (negative impact of updating on availability of data or access performance) are affected. The most obvious measure, therefore, is to restrict the provider's access as much as possible. This is, in turn, contraproductive in the sense that the provider should do the job as beneficial as possible.

If performance and reliability shall be maintained as the established method of FTP incremental updating is in place, LUP tools can still improve the status of the database mirroring, as it allows both quality measuring and success control, which is not provided by the database maintainers due to the restrictions set by FTP server-specific limitations.

5.5 Synchronization

Maintenance of update is not necessarily a matter of confidence in technology. Besides the number of entries, the version numbers of the database may be used to achieve synchronization . Considerations to employ checksums are not feasible due to the amount of data used. E.g., two two- byte checksums for sequence and annotation data section would cause an additional four bytes to be transmitted per entry. We consider it more reasonable to count the entire amount of sequence symbols and use pragmatic tools (e.g., a FASTA run) on the production database to achieve a reasonable functionality testing of the production data.

As the LUP tools inherit the POLL paradigm with feedback in a two-step query (as opposed to a procedure like in FTP where all transactions are achieved in a single session), the synchronization of entries is part of the update method and does not need to be carried separately. This is advantageous as well if the local database is in a different format than the provider database, as long as the lists are compatible in format, usual measures of identity (number of lines, data file size) do not need to be applied. It should be emphasized that these methods will fail if a formatting step in employed; a NBRF formatted set of files will always have different sizes, line numbers etc.

If the FTP method is chosen to obtain incremental updates, LUP tools can be used to integrate the data into the given set of data for an incremental update. However, the synchronization checking might not be available in transparent fashion as if the provider supplied the list of entries like proposed in this mechanism .

5.6 Atomic updates

The calculation of a checksum for transmission is an additional effort which might allow that entries are changed and reconstructed from fragments rather than entire data transmission. This so-called 'atomic' updating may be used to update large sequences with a minor effort by transmitting only the changes of the entry rather than the entire data itself. There are two major arguments why this option is unattractive.

- First, we face the problem of an internal standard. The updating mechanism allows that the customer updates his set as frequently as desired. This might lead to a mismatch of entry versions present in provider and customer databases. As the data are temporary in nature, any atomic updating would require that a defined entry version is changed rather the current version, which might differ between provider and customer. As neither of the two define roll-back history files which would allow to reconstruct versions earlier than the one present in the current version of the database no common basis can be found.

- Secondly, experience shows that only a fraction of the entries is ever updated. Therefore, the calculation, transmission and comparison of checksums is an unnecessary overhead. However, as we do allow multiple queries and contacts, we propose to handle this atomic updating as a separate service: The provider shall prepare a list of entries which is may be updated on atomic basis, and this service is separately handled from the regular updating. If this procedure is applied before the 'full entry' updating, any failed atomic updating would be 'repaired' by the subsequent normal procedure.

Acknowledgements

The staff at the EBI, in particular Jeroen Coppieters, were very supportive and provided helpful discussion. The BioComputing Facility at Basel is funded by Basel University, and received additional grants from the Swiss National Science foundation and the Bundesamt fuer Bildung und Wissenschaften.

References

1. R. Doelz, in: The EMBnet Project, Computer Networks and ISDN Systems **25**, 464 (1993).
2. P. Stoehr, EMBnet PSI-COPY data transfer. Unpublished. (1990).
3. P. Gad, The NDT protocol. Unpublished. (1993).
4. R. Doelz and L. Rosenthaler, The DBTOOLS package. This software was published as 'dbtools' on the bioftp.unibas.ch file server in 1993 and is now superseded by the current version.
5. T. Etzold, and P. Argos, Comput-Appl-Biosci **9**, 49 (1993).
6. R. Doelz, R. Comput-Appl-Biosci **10**, 31 (1994).
7. N. Redaschi, R. Doelz, F. Eggenberger, F. in: HASSLE v5, Dr.U.Doelz Verlag,Basel, 1995.