Data Design Considerations for Distributed Applications

Leigh Solland
Cognos Corporation
2301 E. Lamar Blvd, Suite 416
Arlington, TX 76006

1. Who Cares?

Every company which uses computers is considering, and most are already using, distributed data processing of some sort.

Large companies:

- are connecting remote processors to mainframes
- are decentralizing operational processing
- are building more operation oriented, mission critical systems
- are tying PC's and LAN's together, which have been acquired and developed in isolation
- are standardizing and connecting decentralized processing

Midrange and small companies:

- are building networks of minis, LAN's and PC's.

Why?

- the cost of acquiring and operating mainframes is very high
- the cost of communications for leased lines from mainframes to remote terminals is high and increasing rapidly
- increased redundancy of equipment achieves greater overall uptime, because even if a part of the system is inoperative, the remainder is still working
- reduced redundancy is possible in people, data entry and data, resulting in lower costs, if the distributed system is well designed
- local control is desired wherever local management actually has authority
- hardware and software growth is more modular and more incremental with networks of smaller computers, and therefore not only less expensive but also easier to manage.

existing equipment can often be incorporated into a new network, and used until it is fully depreciated

diverse solutions may be integrated, incorporating turnkey systems, purchased application software packages, and custom development, even when purchased solutions are only available on heterogeneous hardware or software environments

2. What is Distributed Data Processing?

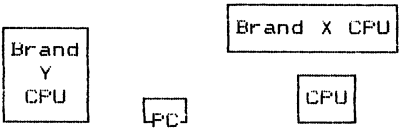
Any or all of the following may be distributed:

the actual data (storage devices and media)
the processing (CPU cycles)
the development effort (programming, report writing)
control (operational decisions)

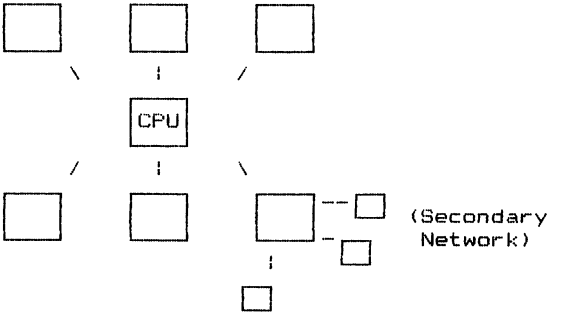
Several configurations are available:

Centralized (one machine, often nearly large enough to perform all processing)

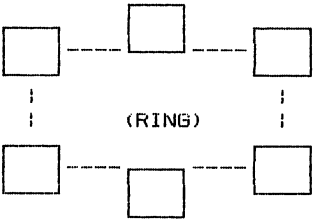
"Distributed" (decentralized authority, with diverse, widely dispersed equipment)



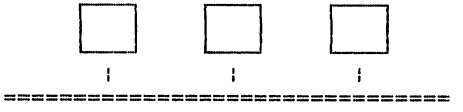
Hierarchical or Vertical (central computer connects to remotes, which may be arranged along geographical, functional or other lines)



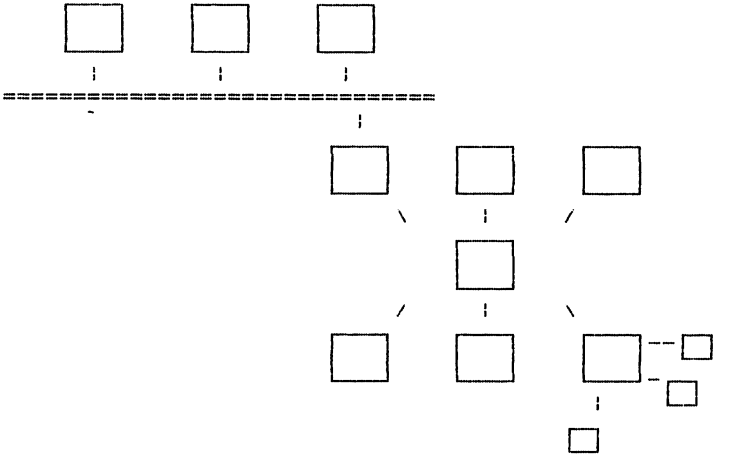
Peer-to-peer or Horizontal (no central computer, but rather a connection which links computers together, such as a LAN)



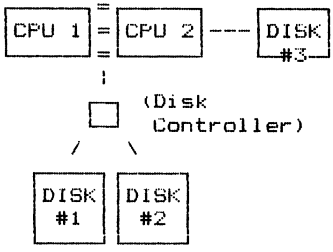
or...



Combinations



Multiple Processors or Clusters (redundancy fosters uptime and reliability, so this configuration is particularly useful at central or otherwise critical nodes)



DDP writing has largely focused on the connection itself, in an attempt to get enough speed and bandwidth to be useful:

- electrical interface (e.g., RS-232)
- asynchronous vs. synchronous communications
- hardware vs. software handshaking
- ISO 7-layer protocols
- DS and NS on the HP3000, other software layers elsewhere
- modem engineering
- LAN's, WAN's and other basic platforms

Less emphasis has been given to the application issues:

- what to distribute (data, processing, development, or control?)
- how to decide?
- how to know when your DDP structure works?!
- how to manage the transition from monolithic or chaotic systems to useful DDP systems?

3. What do we already know about data design?

File design:

one record type per file (or set)
all record items dependent on "the key, the whole key, and nothing but the key, so help me Codd!"

Keys:

fewer keys mean better write performance in Entry/Update
more keys mean better read performance in Inquiry and Reporting, by decreasing expensive serial reads

Locking:

get all locks before you start a transaction
hold the locks for as long, but only as long, as you need them
always get locks in the same order

Tuning goals:

online processing needs to minimize elapsed clock time
batch processing needs to minimize resource consumption

Locking goals:

online processing needs maximum concurrency (tending to shorter transactions)
batch processing needs maximum consistency (tending to longer transactions)

4. What is different in the DDP environment?

Locality issue:

faster response is possible if the data file is local
local processing is insulated from down time on other nodes if the data file is local

Integrity issue:

multiple copies of a file invite unsynchronized changes
multiple input locations require synchronization processing if multiple copies of a file exist and must be updated immediately
there is a tradeoff between locking delays, delays for synchronization of update processing, and out of date information

5. There are only THREE possible designs for a distributed file

Centralized or otherwise shared data:

one copy of file, often in central location
shared access to data of common interest

ADVANTAGES:

easy to update (one place for modifications)
easy to regulate (locking, duplication, data validation)
data is always up to date
fast and easy for analysis and "global" reporting.
good for reference data which changes frequently(e.g., customer credit or accounts receivable check lists or commodity product prices.)
good for data updated often from more than one place
often a very good solution for data which has become "informational", and is no longer of "operational" value.

DISADVANTAGES:

will always be slower to access and update, if located on a remote node, than local data.

Copied (or "cloned") data:

multiple copies of a file
local access to data of common interest

ADVANTAGES:

fast access on reads (always local)
high network reliability (not dependent on any other node)
must be modified in central location and redistributed periodically, or else all modifications must be propagated to every copy throughout the network
best for data which is very static, (e.g., lists of credit terms, branches, state tables, zip code lists) or where updates always come from one place (e.g., supermarket price lists).
could be appropriate for types of data such as customer lists, prices or inventory masters, if they change infrequently.

DISADVANTAGES:

may be very slow on writes, depending on "write-through" requirements to update copies on other nodes.

updates are generally difficult, with locking considerations and deadly embraces over multiple CPU's, and dependence on other nodes' uptime.

may be out of date.

Split data:

local access to data of local interest

this is what many people think of as "distributed" data

ADVANTAGES

very fast and reliable for local data

good for local transactions (orders, production) or detail of interest only to local users, but not needed by the entire organization (directions to drive to customers' offices, local inventory balances)

DISADVANTAGES

slower and potentially less reliable for access to remote data.

application complexity may increase, because each application has to either incorporate the logic to search for remote data or else use an extremely slow brute force serial search of all nodes.

it is very difficult to do analysis or reporting on a basis broader than the local versions of the files. It is usually necessary to combine selected records from each node in a single location first.

the data records are very susceptible to duplication and divergence due to unsynchronized updates.

6. How do you decide which to use?

Who needs to see the data?

consider that reports may be run at remote nodes and sent to management at headquarters
Do they need all the detail? Can it be extracted or summed? (For instance, accounting people like to collect data, but do they really need details on which salesman called on which prospect for the past twenty years?

How do data updates happen?

Who creates or deletes data records?
Who writes, updates or changes data?
How often does the information change?
How soon do the changes need to be available to everyone who has a need to see the data?

How large is the file?

consider both absolute size (bytes, records) and relative size (compared to other data in the system or organization)

What is it used for?

customer inquiries?
monthly G/L batch processing?
weekly management reports?
on-demand management reports?
which manager?

What other constraints must you consider?

communications costs?
response time minimum?
current equipment to be used?
company standards to be dealt with?

What does your ORGANIZATION look like?

model the real world, not the current model!

7. What is ideal?

Guidelines:

READ/WRITE LOCATION:

if data is read by one node only, put it there! ... else ...

if data is read by multiple nodes, but written (created, updated, deleted) by one node only, consider locating it at the writing node ... else ...

if data is written by multiple nodes, consider centralizing it if it is dynamic, or cloning it if it is static.

UPDATE FREQUENCY:

if updates are done seldom or never, multiple copies are fine ... else ...

if updates are frequent, one copy is better.

UPDATE DELAYS:

if delays in seeing updates to a shared file are acceptable, look at capturing local update transactions and batch transferring them to a centrally located, single copy of the file ... else ...

if updates must be immediate (e.g., in a sales inventory), the updatable files must be available on-line and performance is much more of a consideration.

FILE SIZE:

if the number of records in the file is small (e.g., a table of state codes and names), it may easily be cloned ... else ...

if the file is large (e.g., the income tax rolls for the U.S.), it will tend to be centralized for storage requirements if for no other reason.

Look at data in a hierarchy which matches the organization

Put each data item at the level where it needs to be, then make it into records. For example, it may be wise to break the Inventory product master into two files, a central or cloned file containing common information such as description and size, and a split file with local information such as stock balance, pricing, and so on.

Do not accept that head office users have a need to see or change anything in the system. If they have such a need legitimately, it may be possible to set them up as valid users on each remote node.

Consider that, to generate common reports, it is probably a good idea to have all the data required by that report at one place. This may be done in batch, if it is only required on a monthly or annual basis.

For senior management reporting or other analysis, snapshot extraction is a very useful option. A periodic process (daily, weekly, monthly, hourly, or ...) creates a new file for the reporting processes, replacing the current reporting file. A snapshot file eliminates old or unneeded records, summarizes wherever possible, eliminates operational necessities such as status flags, and generally loses unnecessary detail. The snapshot process has a side benefit of keeping data consistent throughout a given time period.

Centralize everything to a level which allows it to be updated by those who have a legitimate need to do so.

Intentionally decentralize for performance reasons only.

Plan to copy and download static information periodically.

Large central files, such as Customers, may be considered too large to copy onto each remote machine. These may be split up according to which location deals with them. Then each node can get a copy of "its own" customer list.)

Watch out for bottlenecks:

if all processes on all nodes depend on a name-and-address master file at the central location, the network is very vulnerable to communication or central node failure.

Be creative:

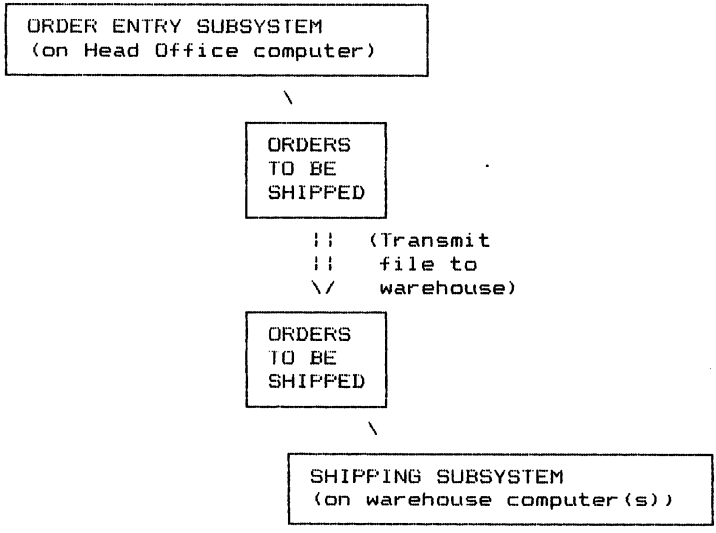
if one or two data items are messing up your design, break them out into their own record or records, and see what happens!

8. What if the ideal is not possible?

Telecommunications Cost:

Millions of dollars per year may be saved by replacing dedicated leased data lines and online terminals with a network based on dialup. The disadvantage is that the files we so carefully placed in the inner layers of the network may no longer be easily available.

Careful design will separate subsystems, which may be running on completely different machines, with common files (i.e., the output of one subsystem is the input to the next). This makes the network transparent to the application analysts and programmers, so that they do not have to consider it any further.



The communications interface between systems then becomes a matter of transferring a file from one system to a lookalike file on the target system.

This interface may then be purchased, developed, tuned or enhanced in isolation from the applications themselves.

Performance Limits:

Network performance may not be satisfactory, no matter how much is spent, if the data is not local.

use audit files to capture transactions, then batch jobs to propagate them around the network. It may be possible to use a file of nodes which need to be updated for each transaction, so that all nodes don't need to be updated for each transaction.

Head Office or other Reporting

use audit files to capture transactions, and batch them up on a periodic basis. The control of the transfer process may be at the remote node or at the center.

reduce the size of what is transferred as much as possible, by use of extraction and summarization.

Intentional Redundancy Requirements

it may be possible, by cloning each remote node's processes and data in a central location, to increase available up time to the remote users. If the local hardware is down, they may still be able to work, using their terminals and the network to access data and programs on the central hardware. I do not feel that this is a particularly effective way to ensure up time, for several reasons: it takes too much work to keep data synchronized, it introduces a need to update both ways for when a disabled remote node comes back up, and these computers are very reliable in the first place!

9. Summary

Make the network model fit the organizational model

- if the structural philosophy of management changes, your network may also have to change. This is known as "the cost of doing business"!

Concentrate on application functions first.

where does management think they happen?
where do they really happen?
where does management want them to happen?

Look at data next.

- put it where it needs to be to accomplish the functions.
- if you have a problem locating data, review your function map: is a function in the wrong place? (Isn't modelling fun?!!"

Then, and not before, look at hardware, baud rates, muxes, phone lines, and the like.

- it may be necessary to modify the model due to costs, communication service availability, or existing investments
- make your compromises here, but BE SURE that senior management is aware of what is going on: they may decide to spend more, reduce a requirement or change a constraint to achieve the overall business goal. Senior management makes business decisions, technical management makes technical decisions. Each of you should let the other do THEIR job!

Start with a pilot

- don't try to implement the entire universe in one shot, or you are doomed to failure before you start.
- get feedback from the installers, users and maintainers of the pilot before proceeding.
- it is not (quite) too late yet to change your mind.

You will never finish this project!

your model, your network and your applications must continually flex to adjust to the changing business climate, management requirements and available new technology.