Tandem TR 85.1

ONE THOUSAND TRANSACTIONS PER SECOND

Jim Gray*, Bob Good**, Dieter Gawlick***,
Pete Homan*, Harald Sammer****

Fall 1984

* Tandem, 19333 Vallco Parkway, Cupertino CA 95014
** Bank of America, Dept 3465, P0 3700, S.F. CA, 94137
*** Amdahl, P0 3470, Sunnyvale, CA, 94088
**** Tandem, P0 560214, 6000 Frankfurt/Main 56, West Germany

ABSTRACT

Several companies intend to provide general-purpose transaction processing systems capable of one thousand transactions per second. This paper surveys the need for such systems and contrasts the approaches being taken by three different groups.

This paper appears in the proceedings of IEEE COMPCON-85, held in San Francisco in February 1985.

TABLE OF CONTENTS

INTRODUCT ION

Most businesses are small, but most white-collar workers are in large organizations. Consequently, most transaction processing systems are small but most transactions are processed by large organizations.

For low-volume transaction processing systems, the critical issue is the cost of designing, installing and operating the system. Improvements in computer and communications price-performance allow the low-end user to adopt brute-force hardware-intensive solutions to design problems. The winners in the low-volume transaction processing business are vendors with application generators or even turnkey systems.

Designers of high-volume transaction processing systems have a different problem. Off-the-shelf data management systems cannot handle current volumes, let alone the volumes projected by the end of the decade. Ordinary transaction processing systems and techniques bottleneck at 50 transactions per second (tps) while high performance transaction processing systems achieve 200 tps. Beyond these transaction rates, users are forced to build their application using ad hoc techniques rather than general-purpose systems.

The airline and travel industry pioneered high-volume transaction processing and currently operates systems at the 800tps range. These systems are built on top of the Airlines Control Program recently renamed the Transaction Processing Facility (TPF]. TPF is an operating system kernel which provides device support for terminals and discs. TPF does not provide a data management system, data integrity, or a high-level programming interface. The tasks of data management, presentation services, network management and transaction management are largely left to the application designer. For these reasons, TPF in its current form is not a complete solution to the thousand tps problem.

Several companies intend to provide general-purpose transaction processing systems capable of a thousand tps. This paper surveys the need for such systems and contrasts the approaches being taken by three different groups.

WHAT'S A TPS

Transaction profiles vary enormously from application to application and even within an application. A large class of applications has staple transactions with the profile:

Read message from network and decode it

Modify 4 database records

Present message to network.

A system's Transactions Per Second (tps) rating is the number of these transactions it can run per second and deliver one-second response time to 95% of the requests.

The price performance of a transaction processing system is also at issue. Ideally, price would include the cost-of-ownership: The cost of designing, programming, and operating the entire system. Unfortunately, it is very hard to define that cost. A poor surrogate for the cost-of-ownership is the five year system price: The five-year price of hardware, software and maintenance paid to vendors. This price is divided by the tps rating of the system to get a price performance metric in the units K$/tps. Typical price performance ranges from 40K$/tps to 500K$/tps.

This metric is bad -- it excludes terminal cost, network cost, programming cost, and operations costs. A low function system will have a better K$/tps because it has low software license fees – yet that system will have high programming and operations costs! Unfortunately, K$/tps is tangible while the cost-of-ownership is not. Hence systems are sold (or not sold) based on K$/tps -- that is a sad fact.

More exact definitions of tps and K$/tps are found in [AnonEtAl].

WHO NEEDS 1KTPS?

Airlines are the most obvious candidates for a general-purpose lKtps system. They have major investments in their current systems but the industry is changing rapidly. Their current software is written in assembly language against a primitive interface. This implies high software costs for providing new services to travel agents, supporting smarter terminals, and networking to other industries. Current systems are difficult to (re)configure and operate. If someone could provide a general-purpose transaction processing system that had a better programming interface and yet had price performance comparable to TPF, then the airlines would seriously consider switching to a new system.

A major issue is the cost and feasibility of migrating from an existing system to a new one. Airlines want to preserve their investment in existing systems. In addition, these systems must always be up, so the migration must be gradual. This poses real challenges to any new hardware or software system.

The telecommunications industry is in a similar state. Telecommunications is getting much more complex. Accounting procedures are changing to include billing by call, by packet, by duration, or even reverse billing. New services such as autodial, call forwarding, conferencing, and so on, are adding to the complexity of call setup and teardown.

These changes in billing and use are making the typical call setup and tear down have a profile similar to the standard transaction mentioned in the last section. The largest switches are running at 200 to 400 tps today. The next generation switches need to have higher capacity as well as have more sophisticated call management logic. Higher speed links will drive transaction volumes to the lktps range.

Time buffered voice mail and electronic mail are classic transaction processing applications. Implementing them on the current electronic switching equipment is unthinkable. Projections for electronic mail transaction rates are astronomical.

As with the airlines industry, existing phone companies (as opposed to new ventures) are locked into an industry-specific hardware-software solution that presents the same conversion vs. evolution dilemmas faced by the airlines. New long haul carriers are projecting aggregate rates of l5Ktps for the network and 2ktps for the largest nodes by 1990 (one such contract was recently awarded). These customers are interested in using general-purpose hardware because they want to benefit from technology changes and do not want to write and support special purpose operating systems and data management systems.

Finance presents an interesting contrast. Banks are interested in converting to online bookkeeping to reduce operating costs. Most banks are a batch operation with on-line data capture and overnight batch runs to do account processing. The conversion to online operation offers the opportunity to rewrite the application using general-purpose systems. This is in contrast to the established travel and telecommunications industries that are already online and hence are not eager for a massive rewrite. However, the problem of smoothly migrating to the new system is still a major issue -- it must be done incrementally.

There is significant advantage to having an online accounting system that ties into point-of-sale networks, automated clearinghouses and automated payment systems. Banks hope to reduce their dependence on paper, buildings, and manual operation by offering their customers a large array of electronic financial services. This strategy pivots on the ability to handle large transaction volumes.

Many European, American and Japanese banks have peak batch rates in excess of 500tps. This workload will shift from the night to the daytime and will grow with the diversity of banking services being contemplated. In addition, the transition from cash to plastic will multiply transaction volumes -- a single 100$ ATM cash debit could become five 20$ electronic POS transactions. Large banks are aiming for open-ended capacity with an initial target of lktps. Several financial institutions have declared their intention to be the lowest cost transaction delivery system in the industry.

Finance is the most promising arena for the development of general-purpose lktps systems. Batch will continue to be a major part of transaction processing for these customers, but it will not be compressed into an 8-hour overnight batch window. Rather, it will be structured as many mini-batch transactions that run as a part of normal processing. Gawlick is skeptical of this statement; he believes that higher transaction volumes generate even larger batch runs. This has been the IMS Fast Path experience.

Several other industries, Government (the biggest business there is), manufacturing and process control all would benefit from a general-purpose lktps system.

Clearly many industries are projecting or have the potential for requiring a lktps system. In the past, these requirements have been met with roll-your-own (RYO) systems. Can general-purpose systems provide this level of performance? This is both a technical and a financial question.

From the technical standpoint, there are no lktps general-purpose systems today. We believe that such systems will be available by the end of the decade.

This leaves the financial issue -- the relative cost-of-ownership of a low-function system requiring significant user extensions (RYO) versus a full-function system maintained by a vendor. The tradeoffs are:

1. The poor k$/tps of general-purpose systems
2. The savings in development and support staff gained by using a general-purpose system, and
3. The trends of these factors over time

We estimate the RYO approach (TFP) price performance to be about 40K$/tps (without x.25, presentation services, and transaction recovery). General-purpose systems have costs ranging upwards from 60K$/tps. This comes to 20M$ added cost over five years for a general- purpose system at a 1KTPs, i.e. 4M$/year. To be attractive, general-purpose solutions must save at least 4M$/year in development and support costs. This amounts to a lot of staff -- about 40 people: 20 programmers, 15 test and pubs and 5 overhead.

It is hard to prove a general-purpose system can reduce staff by 40 people. The arguments are mushy: Productivity, features, skill requirements, management structure, and company politics all play a role.

On the other hand, estimating cost trends over time is easy. Hardware costs drop by a factor of two every few years. Personnel costs continue to rise. By the end of the decade, the relative costs may well be 20K$/tps vs 30K$/tps. At this price, the marginal cost for a general-purpose system is 2M$/year -- about 15 people in 1990 dollars. It is just a matter of time until general-purpose is more attractive than an RYO.

The current attractiveness of a general-purpose solution depends on the cost of ownership, the additional risk of doing an RYO solution, the time before the RYO and general-purpose cost curves intersect, the expected lifetime of the application, and the degree to which two approaches meet the technical requirements outlined below.

TECHNICAL REQUI REMENTS

A lktps system is large. To begin, the machine room has 50M$ worth of hardware in it -- upwards of 100MIPs of processor, over 100 gigabytes of data on 500 discs.

If one assumes a 50 second average think time, the system supports 50 thousand active terminals (at 2K$/terminal this is 100M$). The corresponding communication network is enormous -- at least 2Mbit of communications lines will converge on the data center.

Such a system must be:

- Manageable
- Available
- Centralized/Distributed
- Granular
- Growable
- Changeable
- Cheap

MANAGEABLE: The system must provide good development and operations tools. For development, this means a high-level programming interface to a data management system and to a data communications system. The system should relieve the programmer of the issues of data representation, either on the terminal (presentation services) or on disc (data management). In addition, it should handle transaction management (fault tolerance). It must provide tools to aid design, development, integration, test, quality assurance and performance assurance.

Operating a system of this size is a challenging job. Any system with this many parts will be in constant flux. Trouble reports for terminals will arrive at the rate of one per minute. At least one communication line will be out at any time. Network management is a giant inventory control problem. Each day, one of the 500 discs will be having trouble. Someone will have installed a new program that has some quirks. Facilities to capture, diagnose, report and track problems must be built into the system.

AVAILABLE: In each of the intended applications, an outage translates directly to lost revenues. Typical requirements allow a single five-minute outage per year. Telephony wants three minutes per forty years.

These availability requirements are achievable with current fault tolerant systems. Unfortunately, operations errors and software errors are the major source of system failures. This is the reason for wanting a high-level programming interface and a general-purpose solution. Great care must be taken in writing, testing and installing software and in managing the system.

In addition, a standby data center is required in case of natural disaster or sabotage. The switchover time is generally stated at 5 minutes. The two data centers must be at least 100 miles apart.

CENTRALIZED/DISTRIBUTED: The system should allow centralized or decentralized deployment without application change. It is very attractive to have local, regional and perhaps divisional design in which the data and applications are distributed among a collection of nodes. Geographic distribution reduces communications costs if there is strong geographic locality of reference and it also provides the basis for disaster recovery.

GRANULAR: A lktps system is very large. It must be decomposed into small units that are units of design, control, performance and failure. A system composed of a few large lumps will have poor characteristics.

GROWABLE: A consequence of geographic distribution is that some of the nodes may process only 10tps while others do lktps. The hardware and software must economically scale from l0tps to lktps. This growth should be non-disruptive.

CHANGEABLE: Addition and movement of devices (discs and communications lines) should not disrupt service. Database and application installation must not disrupt service. Database reorganization and repair and application repair must not disrupt service. Good online testing, measurement, and tuning tools are required as part of change management.

CHEAP: The cost/tps must be competitive -- 60k$/tps is acceptable, 40k$/tps is desirable. The cost/tps should be stable in the range l0tps to l000tps.

THREE APPROACHES TO 1KTPS

No system meets all these requirements. Some meet them by degrees. The three approaches being taken to this problem attempt to evolve current systems -- IMS, Tandem and TPF. At least one of the authors believes each system is now capable of 500tps.

IMS: IMS-Fast Path is IBM's flagship general-purpose high-performance transaction processing system. The largest IMS Fast-Path [IMS] installation peaks at 180 tps. Advances in processor speeds and memory sizes, along with elimination of IMS cpu and storage bottlenecks should allow IMS Fast-Path to process lktps and support 50k terminals by the end of the decade. The 370 architecture has good granularity from the 4341 on up. The weaknesses of IMS are manageability (MVS-VTAM- ... are complex), availability, centralization, changeability (most changes to IMS, MVS and VTAM are done off line), and cost. We believe IMS developers are addressing each of these issues.

Tandem: The Tandem architecture comes close to satisfying the requirements [Tandem]. It provides a good development environment, has high availability, can be centralized or distributed, and has linear growth. Applications and databases can be added online and online maintenance is allowed. The largest existing system peaks at 60tps but a 150tps system has been benchmarked. Its weaknesses are network management, online installation of new hardware, and cost. Each of these areas is being addressed. A non-technical issue is that Tandem is not as well established as IBM.

TPF: An alternative approach for the 370 is to enhance TPF to have a data management system, a network management system, a transaction management system, and so on. TPF is capable of processing 800 tps with 60K terminals today (such systems exist). A sizeable effort at Bank of America plans to use TPF as the basis for a high-volume transaction processor front-end that passes complex transactions on to IMS back-ends. BofA is enhancing TPF to have message-based requestor-server architecture. The message system will be the basis for distribution. In addition, they are enhancing a data management system (ACP/DB) to have transaction logging and media recovery. It will include logging for message integrity. The system will clearly satisfy the requirements of distribution, granularity, and cheapness. Meeting the other requirements (manageable, disaster recovery, and changeable) is less clear.

TECHNIQUES

Each of these projects can be successful or fail. The technical problems are solvable; the major causes of failure will be non technical (management, politics, change of direction ...).

Gawlick [Gawlick] outlined the IMS Fast Path approach to the technical problems. Here we sketch the approach being taken by Tandem and the TPF groups.

The key to fault tolerant and distributed execution is to adopt. a message-based interface among software modules. It gives granularity of software and hardware modules. Automatic transaction logging ties together the execution of these modules into atomic and durable units of application work. This approach is provided by Tandem, CICS/ISC, and is being added to TPF by the Bank of America developers.

Given this basis, one can implement the requestor-server architecture that allows functions to be added, moved or cloned for throughput, availability and reliability. It allows modular growth of processors and devices as a mechanism of scaling up the system.

Rather than having one or two large processors, one can have many smaller processors. Granularity is the major advantage of this approach. Multiple processors provide granularity of growth (one can add 10% by adding a processor to a group of 10) and granularity of failure (one loses only 10% when one fails).

The key to fault tolerant storage is replication of data on media with independent failure modes and the use of transaction logs to redo or undo transactions in case of failure. Both Tandem and TPF provide duplexed data. Transaction logging is supported by Tandem and being added to TPF.

Disaster recovery is provided by replicating the data at a remote site and spooling the recovery log to it. We imagine two (or more) data centers each capable of 500tps. Each site provides disaster recovery for the other as well as providing service to half the customer base. Disaster recovery uses about 10% of its capacity to apply the log to the database. Switchover is possible in seconds. This technique is currently being used by Tandem customers -- (at lktps, the log data rate is .5mbyte/s).

At lktps, logging needs to be streamlined to do group commit as outlined by [Gawlick].

One key to high performance is the reduction of disc IO associated with transactions and the conversion of random IO to sequential IO. Disc IO wait times are a major contributor to transaction service time and disc arms are a major contributor to system cost unless the system is carefully designed. Random IO to discs runs at 20 IO/s. For 4kb blocks this is a data rate of 80kb/s. Current discs are capable of 3mb/s. By converting from random to sequential IO, disc utilizations improve by a factor of 10.

Streamlined logging techniques imply more than 0.1 log IO per transaction. By using large main memories, random IO on high-traffic data can be converted to sequential IO as follows. The data resides in main memory. All updates to the data are written to a sequential log and retained in main memory. Periodically, a fuzzy dump of the data is written to secondary storage as a sequential scan of the main memory database. This has converted all random to sequential IO. IMS Fast Path MSDBs use this technique [Gawlick].

IMS Fast path has techniques for dealing with hot spots in the database [Gawlick]. Another approach to hot spots is to design around them. For example, if the end of a sequential file is a hot spot, convert it to a B-tree or split it into several sequential files. The history file of the DebitCredit benchmark gives an example of this. One can have a history per branch, or even a history per account.

When hot spots cannot be avoided, then the field calls of IMS Fast Path must be used. One reservation about these techniques is their complexity for programmers to understand and their limitations for distributed systems. Putzolu points out that an SQL interface could hide this complexity [Putzolu].

Lastly, there is the problem of batch processing. If the data is not structured correctly, periodic processing requires huge sorts and computations. For example, monthly billing passes over each transaction on an account and produces an itemized list. If this requires a sort of the transaction history file each night, there will be a huge batch requirement for a lktps system. An alternative is to do the batch work as a normal part of transaction processing. In the latter design, the account history is stored sequentially next to the account. It is available for online inquiry. The mini-batch run fetches the account and its history as a sequential operation without any sorting being required.

A lKtps system has a peak-day average of 300tps and an average day average of 200tps -- but the system is sized for peak demand. This means there is substantial processing capacity at off-peak hours. This spare capacity allows each of the dual 500tps disaster recovery sites to take over the workload of the other if the disaster happens off-peak. This extra capacity is also available to run the mini-batch (and maxi-batch?) operations during periods of light load. Gawlick, for one, is skeptical of this; he believes batch will be with us for a long time and may dominate system sizing.

Summarizing the last few paragraphs, we propose to avoid hot spots and batch processing by application design.

SUMMARY

There is a clear need for high volume transaction processing systems. Several groups are attempting to evolve current systems to meet that need. Most of these efforts are two years old and will not bear fruit for another two or three years. It seems likely that at least one of them will succeed in producing a system that meets most of the requirements of being manageable, available distributed, granular, growable, changeable, and cheap.

REFERENCES

[AnonEtAl] "A Measurement of Transaction Processing Performance," Tandem TR 85.2 Tandem Computers. 1985

[Gawlick] "Processing Hot Spots In High Performance Systems", Gawlick, D., Proceedings of IEEE Compcon, Feb. 1985.

[Putzolu], Putzolo, Gian Franco, private communication.

[IMS] "IMS/VS Version 1 General Information Manual" IBM form GH20-1260-12, 1984

[Tandem] "Transaction Monitoring in Encompass, Reliable Distributed Transaction Processing", Borr, A.J., Proc. VLDB-7, ACM #471810, 1981

[TPF] "ACP/TPF Concepts and Architecture", IBM form GH20-2157, 1984.