Jim Gray's Tandem Contributions

John Nauman

john@johnnauman.com

Wendy Bartlett
Hewlett-Packard
19333 Vallco Parkway
Cupertino, CA 95014
+1 408-285-6262
wendy.bartlett@hp.com

ABSTRACT

This paper discusses the contributions and accomplishments of Jim Gray while he was at Tandem Computers between 1980 and the early 1990s.

1. INTRODUCTION

In 1980, the year that Jim joined Tandem, the world was a different place. Ronald Reagan was elected President. John Lennon was shot in New York. Smallpox was eradicated. The Best Picture Academy Award went to "Ordinary People". The Grammy for Best song went to The Doobie Brothers for "What a Fool Believes". The New York Times number one hardback fiction work was Sophie's Choice. In the computer industry Microsoft approached IBM to develop BASIC for its personal computer product - the IBM PC. Xerox, Digital Equipment, and Intel jointly announce the Ethernet network specification. The U.S. led in IC production with 72% market share, Japan had 16%, Europe had 6% and the rest of the world produced 6%. For the first time, the total number of computers in the U.S. exceeded one million units.

It was a long time ago, but many of the things that were industry issues were also critical to the development of the world we know today: the notion of independent benchmarks to allow customers to determine the system which best fit their needs; the concept of transactions as atomic units of work which either happen or don't; the need for a more robust and functional way to access data; an understanding of why computers fail and what can be done about it; the concept of decentralization/scalability and how it can actually be realized. Jim was intimately involved in each of these during his time at Tandem and his technical insights and contributions will be explored. But Jim was also a member of the Tandem community and as such was influenced by, and had a significant influence on, the company's culture and products.

2. GETTING TO TANDEM

When Jim joined Tandem, in 1980, it was about a $100M company with an unusual (and somewhat prescient) culture which has been adopted and adapted by many startup companies since. It was energetic, driven, had good, often unique, ideas and approaches and above all, from Jim's perspective, smart people.

The Tandem product consisted of hardware and software that worked in concert to provide the first real commercial example of fault-tolerant computing. Called the NonStop I, it consisted of independent, proprietary processors that communicated over a high-speed bus - allowing a primary process running in one processor to regularly report its status to a backup process in an independent processor. The underlying operating system (Guardian) was built using these capabilities to allow recovery from errors at the system level. The I/O system included parallel, independent I/O controllers and software processes which used the same process-pair mechanism to ensure I/O was accomplished. In the event of the failure of a process or processor the backup would take over and complete the task. The product included a basic programming language (Tandem Application Language (TAL)), COBOL, a data management system (ENCOMPASS) which included data access (ENSCRIBE) and query (ENFORM) capabilities, as well as a high level terminal/screen interface, Screen COBOL (SCOBOL).

3. JIM'S TENURE

During Jim's time at Tandem the system and software evolved significantly, frequently because of Jim's influence. But anyone who knew Jim - before, during or after Tandem - will realize that his influence extended far beyond the area of technical contribution. Jim worked to refine and expand Tandem as a company as well as its products. This could be most readily seen in his interactions with customers and with the sales force. To ensure he made the best impression on prospective Tandem clients, Jim kept a suit hanging on the back of his office door. If someone needed a technical spokesperson to address a customer's concerns, Jim could transform himself from a dressed-down engineer/architect to a super-product-manager in a matter of minutes. In this role, he helped many Tandem customers and prospects understand where Tandem was headed and why.

During Jim's time at Tandem the product line was continuously and impressively extended. On the hardware side the company went through several processor and I/O generations. While Jim was not involved in the direct design of these products he played a key role as a sounding board and also provided input on how this evolution would impact the software products. On the software front, Jim had much more direct involvement. He worked to define and realize the Transaction Management Facility (TMF), guaranteeing consistent transactions. Jim was also involved in the production of ENABLE - a high level software product which generated fault tolerant applications - producing everything from the user screen interface to the application logic to the database access. Jim was integrally involved in the development of NonStop SQL, a version SQL which significantly enhanced ease of database access and guaranteed fault tolerance in the transaction environment.

Jim had an ongoing interest in tools that make (accurate) data readily available for use. This interest benefited not only Tandem's customers but also Tandem's employees – one of Jim's first creations was TELE, an internal program that made the employee database accessible to everyone at Tandem. In addition

to displaying employee phone numbers and org charts, if you knew the right incantation then TELE would display additional interesting items such as employee job codes.

While Jim was an active participant in most of the software products developed by Tandem during his tenure, his major influence was felt in his ability to attract smart and highly-motivated people to work on these projects. No one who worked with Jim on any project at Tandem will forget the excitement and innovation he brought to the work that was going on.

During his tenure at Tandem Jim changed from a superb architect and software engineer to someone who also understood the development process, the requirement to listen, comprehend, and respond to customer requirements, and how to help to make a business successful.

In addition to his development work at Tandem, Jim found important "problem" areas of computing and published papers that broke ground or expanded the understanding in many ways. Jim's Tandem legacy includes an array of technical reports on a wide range of topics.

4. TRANSACTION PERFORMANCE

Probably the most visible area that Jim contributed to was that of measuring the performance of the transaction processing systems of the day. This actually began during Jim's work at IBM (on the Future System (FS) and DB2 projects) when performance was discussed in terms of Sort, Batch and Interactive applications. Standardized measures would allow manufacturers to benchmark their system against both their own and competitive solutions and to enable users to compare the various products. The systems of the day were largely mainframes and mini-computers. Existing measures (MIPS, Whetstones, MegaFLOPS) all focused on CPU speed, not application or user observed performance. Jim felt this had to change.

At that time the highest-performance interactive transaction systems were operated by airlines. The system in operation was called the Airline Control Program (ACP) and later Transaction Processing Facility (TPF). This system ran on IBM hardware and was ruthlessly optimized for performance. Since applications were coded in machine language, and focused exclusively on performance, the running joke was that you could write any application you wanted as long as the program was not more than 200 bytes. This system had a transaction rate of around 800 transactions per second (tps), but had extremely constrained and low level user and data interfaces. IBM's Information Management System (IMS) Fastpath was able to reach 180 tps with somewhat higher level interfaces. The shortcoming of both these systems was their very poor extensibility, fault tolerance and, in some cases, data integrity. Each of these systems, as well as others, would claim high transaction rates - but this was an apples to oranges comparison. The transactions did different things. Jim and others realized that, for there to meaningful comparison between systems, a transaction benchmark was required. The first "accepted" transaction profile developed was based on the Debit/Credit (or Teller Cash Drawer) concept. A customer interaction with a bank teller (or ATM) could be reduced to a few messages (input and output), the disbursement or acceptance of funds and the updates to the customer's account to reflect the activity. This transaction evolved to be called TP1 (or ET1) and was used by Jim as he explored transaction performance in papers, lectures and product design over the next ten to fifteen years.

Jim began by discussing things which were important to customers but were sometimes hard or impossible to measure at the time (Cost of Ownership (COO), thousands of dollars in cost per transaction per second ($K/tps), etc.). Looking forward it was clear that, as more processing moved to interactive transactions, the level of the interface would need to move up to allow faster and less expensive development. Jim's original strawman goal for transaction processing was that 95% should have a response time within one second. In addition, the best-of-class system would be manageable (development and operations tools), available (5 minutes of downtime per year), granular (to allow the system to be decomposed for reliability or manageability), "growable" (different nodes might require different initial sizes and might grow at different rates), changeable (things change) , and "cheap" (at the time). His papers in 1984 ("One Thousand Transactions per Second") and in 1985 ("A Measure of Transaction Processing Power") presented the first well thought out discussion of what transaction performance was, why it was important and how to measure it. This concept and the TP1 benchmark were the precursor to today's Transaction Processing Performance Council and the set of benchmarks which are universally viewed as the standard for online transaction processing.

5. INDUSTRIAL STRENGTH SQL

During Jim's tenure at IBM's San Jose Research Division he was heavily involved with the System R project and was integral in moving the System R concepts into products, at IBM and elsewhere. Clearly the most recognizable of these was SQL, the interface to the relational data model. First developed by Ted Codd, the relational model of data access was incorporated into System R. The access method, Structured Query Language (SEQUEL and later SQL), for the most part was viewed as nothing more than a "toy". From System R, Jim moved to the IBM development group responsible for the DB2 project. When he moved from IBM to Tandem, Jim was convinced that a relational approach to data and SQL as the access language were critical to allowing the database development environment to reach a cost-effective point. Prior to the NonStop SQL implementation most vendors viewed SQL as an information center or productivity tool. They typically provided a non-SQL interface for the "industrial strength" applications. Jim, on the other hand, felt that SQL had to be the core means of database access. He was convinced that the relational model, as expressed in SQL, was critical to moving forward on the interactive application front. Its ease of use characteristics and structure made it something developers could utilize to conceive, design and build applications quickly. It also allowed these applications to be understood easily over time, allowing maintenance and extension.

Tribute to Honor Jim Gray, May31, 2008, Berkeley, CA, USA.

The design tenets Jim felt critical were:

1) To be integrated with the Tandem networking and transaction processing system.

2) To provide NonStop access to data - in case of a fault, only the affected transactions are aborted and restarted: data is never unavailable.

3) To support modular hardware growth, and as a consequence support tens of processors executing hundreds of transactions per second.

4) To allow data and execution to be distributed in both local and long-haul networks.

This was a very ambitious project on several levels. The application program was protected by transaction locking and logging. In addition, all device drivers and system processes ran in NonStop mode so that they could tolerate any single hardware failure and many software failures without disrupting service. The disk processes maintained mirrored disks so that a disk failure would not disrupt service. If a process or processor failed all the transactions being run by that process or processor were aborted (rolled-back); but unrelated transactions were unaffected by the failure. On failure, ownership of all devices (disks and communications lines) was automatically switched to other processors.

First, the sheer technical challenge of providing a fault tolerant and high-performance transaction processing system with SQL as its interface meant that much of the underlying Tandem data management structure needed to be reworked. Secondly there was the ongoing question within the company of whether this was the right direction and would provide sufficient differentiation and benefit to justify the cost. Not surprisingly, Tandem customers and prospects were both impressed by and strongly supportive of the effort - since they well understood its value in simplicity, functionality and future maintainability. NonStop SQL was developed by a relatively small team, many of whom Jim recruited from outside Tandem. He served as everything from architect to developer to cheerleader within the team while at the same time continuing to explain the benefits to Tandem's upper management and fostering customer interest and support. While there were implementations of SQL both before and after NonStop SQL, none were integrated into the underlying system, as well as being fault tolerant and expandable.

A Google search of SQL today returns 248,000,000 hits. Current SQL products range from IBMs DB2 product to the Microsoft SQL Server. Jim's ability to look at a problem like the coming need for faster, more capable and easier to maintain transaction processing systems and see how a new data model, language and system could help realize that is, in no small part, a reason for SQL's acceptance and prevalence in today's environment.

6. WHY COMPUTERS FAIL

Not surprisingly, Jim spent a lot of time at Tandem investigating why computers fail and what can be done about it. He characterized this as "Reliability and availability are different: Availability is doing the right thing within the specified response time. Reliability is not doing the wrong thing." Up until this time, there had been academic work on computer failure but nothing that really addressed the commercial fault-tolerant system.

One of Tandem's strengths was the fault tolerant nature of the hardware and software design. Jim realized quickly that designing hardware to be fault tolerant, though tricky, was the easier part of the problem. He worked with a number of Tandem customers to explore the failures they encountered and his results were enlightening and somewhat unexpected. Hardware "mean time between failure" (MTBF) was measurable in years. Failures occurred in several areas in running systems. First, there is "infant mortality" which consisted of new hardware or software still having the bugs shaken out. This accounted for about 30% of the failures. Systems administration, including operator, configuration and maintenance actions accounted for more than 40% of the failures. Software faults accounted for about 25%. Finally there were environmental failures, which accounted for a relatively small percentage of outages.

As Jim stated, "The implications of these statistics are clear: the key to high-availability is tolerating operations and software faults". However, a secondary implication of the statistics is actually contradictory. New and changing systems have higher failure rates. Therefore a way to improve availability is to install proven hardware, and software, and then leave it alone. On the other hand, a high percentage of outages were caused by "known" hardware or software bugs, which had fixes available, but the fixes were not yet installed in the failing system. This suggested that one should install software and hardware fixes as soon as possible. The conflict: "never change the system and do the changes ASAP".

Since failures happen for many different reasons one of the key points in fault tolerance is to contain the failure and its impact as much as possible. To that end Tandem added a Transaction Management Facility (software) to its fault tolerant system (NonStop and Guardian) to ensure that "units of work" either completed or, in the event of a failure of some sort, removed their effects completely. Due to the unreliable nature of data communications at the time Jim also proposed redundant, resumable communications sessions. Finally, he noted that dealing with system configuration, operations, and maintenance remained an unsolved problem. Administration and maintenance people were doing a much better job than we had reason to expect. Since we couldn't hope for better people, the only hope was (and still is) to simplify and reduce human intervention in these aspects of the system.

7. DECENTRALIZATION / SCALABILITY

Another area that Jim probed while at Tandem was that of decentralization, scalability and particularly distributed processing. Tandem systems were designed to be scalable – growing from 2 to 16 processors in a single system, with possibly hundreds of connected systems. In the 1980's the telephone was the best example of a distributed, decentralized system. It contained thousands of computers, and almost a billion terminals. Today the best example is probably the internet.

One of the issues that troubled Jim was that most of the 1970's had been spent trying to develop centralized data and applications leading to an "integrated database". Tandem's goal, on the other hand, was to allow customers to use a decentralized system (many computer systems networked to provide a common service). Clearly there is no "best" system design, but Jim noted several areas which needed to be considered: capacity, response-time, availability, cost, security, and modularity.

The main technical problem unique to decentralized systems is the lack of global (centralized) knowledge. Where are things?

Unfortunately, security, integrity, auditability, performance and changeability all are adversely impacted by having a centralized system. It seemed likely that the concept of an integrated database would be restricted to single computers or at most local networks of computers managed by a single authority.

Decentralized systems, by necessity, must do more communication, and while the communications overhead and availability necessary to realize such a system were not insignificant, they could be minimized and the overall benefit could be significant. Decentralized systems lend themselves strongly to the ”requestor-server” approach, which in turn can lead to a more robust and resilient system in areas like availability through fault tolerance, parallel processing, modular growth and geographic distribution (putting data near its consumers).

In a decentralized system, the time and equipment cost to transport messages rose by at least an order of magnitude at each degree of distribution and the reliability of message transmission dropped by at least an order of magnitude at each degree of distribution. As a consequence, the message cost of a distributed algorithm was an important measure of its cost - messages were expensive and wide-area messages were very expensive.

So at the time, the case against distributed systems seemed pretty clear: wide-area networks were slow and unreliable - not to mention fabulously expensive. Local networks were thousands of times faster, more reliable, and cheaper. However, in comparison to centralized systems, local networks waste instructions and time. The narrow focus on message cost ignored the benefits of distributed systems - allowing many processors to be applied to a problem in parallel. Distributed systems offer high availability through fault tolerance. They can also provide modular growth, and geographic distribution of data - putting the data and processing next to the user. Distributed systems offered good peak performance through parallelism, and good price performance by using inexpensive components. As a final point, Jim submitted that, often it was simply not possible to construct a shared memory system with comparable power.

While there will probably never be a clear winner in the centralized/decentralized processing debate, Jim’s work foreshadowed much of the decentralization and distribution we see in today’s systems.

8. CONCLUSION

Throughout his career, and particularly while he was at Tandem, Jim recognized areas where the existing ideas and practices would not be sufficient in the future. He became involved in those areas from both a theoretical and practical perspective and moved them forward with insight, research, papers, presentations and products.

9. REFERENCES

Rather than list individual references to the above information the reader is directed to a catalog of Jim’s papers and presentations at:

http://research.microsoft.com/~Gray/

and select “Publications” from the left column.