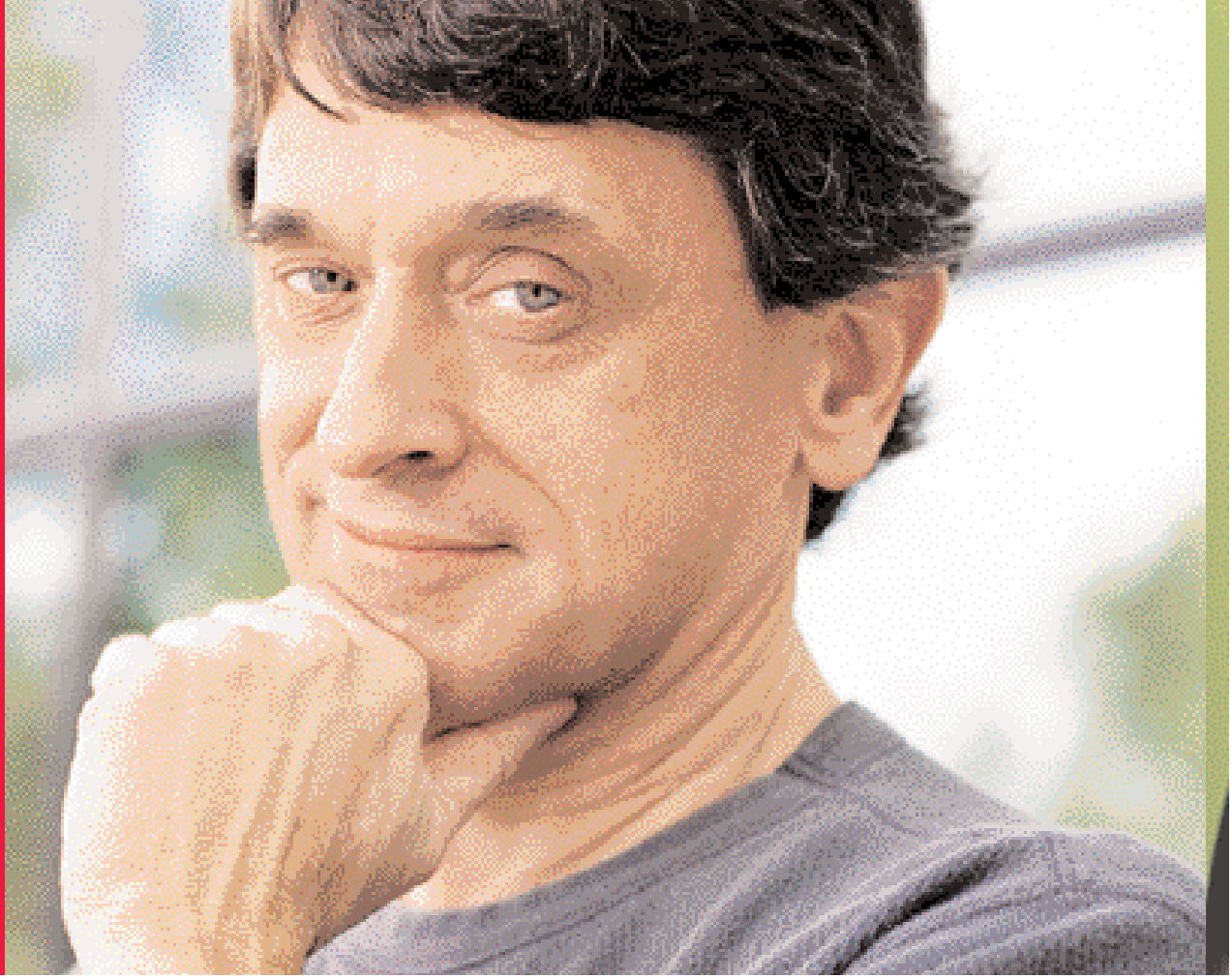

Charles Johnson and *NonStop* TMF software go back many years.

Boosting performance with every transaction

A NonStop system architect discusses the importance of data management

Charles Johnson practically cut his teeth on Transaction Monitoring Facility (TMF) software for *NonStop*™ systems. He worked with TMF1 in the early 1980s as a consultant. In 1989, he joined the former *Tandem*™ TMF group and was intimately involved in the development of TMF2 and TMF3, particularly in performance-related areas. He has received two U.S. patents for TMF work, and seven more are pending. "I'm the oldest TMF developer around," he said wistfully, but with a touch of pride. The product was renamed *NonStop* Transaction Manager/MP for several years—but to Charles Johnson and other old-timers, it was still TMF. With a nod to tradition, the product is now officially called *NonStop* Transaction Management Facility, or *NonStop* TMF.

In the following interview, Johnson explains what this powerful (but little-understood) product does and the enormous benefits it delivers for Compaq *NonStop Himalaya*™ system users.

Q: What is *NonStop* TMF?

A: *NonStop* TMF software is the facility that manages the interrelationship between all the fundamental elements of a *NonStop Himalaya* system—things like transactions, locking, database consistency, log serialization, fault tolerance, availability, scalability, and data integrity. These are the elements that fit together to make a consistent computing outcome for the user.

You can really think of *NonStop* TMF as being a "truth management facility," because we make sure the database correctly records the truth that the

users have told it. When they begin a transaction, update the database, and then end that transaction, and we say "OK," we have to ensure that our OK sticks, even if there's a system failure. When the system comes back up, that transaction has to be accurately represented in the database and interleaved with everyone else's database work.

Q: So *NonStop* TMF plays a big part in ensuring database consistency?

A: Absolutely. In the database world, people talk about ACID properties. ACID stands for atomicity, consistency, isolation, and durability. These are the properties of good databases, and *NonStop* TMF ensures that *NonStop* system databases have all of these properties.

Transaction atomicity means all or nothing. A transaction either happens or it doesn't—it never partially happens. With *NonStop* TMF, if your process fails halfway through a transaction, the transaction is aborted, and the database returns to its state at the beginning of that transaction. You don't have to remember a thing, because *NonStop* TMF remembers all of it. Without *NonStop* TMF software, if a process fails, you have to figure out what the state of the database is—and then somehow set it right. This is very hard, because the database can have a nearly infinite number of states.

Consistency refers to log serialization, because the real database is in the log. If there are multiple nodes, all those logs have to interlock. This is done in *NonStop* TMF with a protocol called *two-phase commit* for network commitment of transactions.

The nice thing about *NonStop* TMF transaction consistency is the all-or-nothing atomicity property—if I don't get it all done and I die, the database goes back to where it was when I started the transaction. Not just the database here, but the database in Cucamonga or Outer Mongolia. I could have transactions spread across the globe, dealing with databases in all different kinds of systems. With *NonStop* TMF transaction protection, it all goes back to the initial state.

Q: This is getting awfully technical. Can't you talk about *NonStop* TMF in simple terms?

A: Not really. In *NonStop* TMF, if you don't talk about deep issues, you don't get to talk about much. The good news is that *NonStop* TMF will handle the most technical, complex details of transaction processing, so you and your programmers don't have to. *NonStop* TMF is at the heart of the system, and at the heart of every other system that's connected to it. It guarantees that the truth of the database across the entire computing space is always consistent. It makes sure the system runs fast, has good performance, can tolerate failures, and can stay up all the time. The issues are very complex, and that's why people don't talk about *NonStop* TMF much.

With *NonStop* TMF, you have these things called transactions—and that's essentially all you get to say that's simple. Transactions begin and they end. But as soon as you start talking about their properties and what they guarantee, then the conversation gets technical. They guarantee atomicity, consistency, isolation, and durability in *NonStop* system databases.

Q: Okay then, can we get back to ACID?

A: We've already talked about atomicity and consistency. The next thing is isolation. Isolation

means when I'm updating something, no one else can read it until I'm done. Or if I'm reading it, no one else can update it until I'm done. That's called locking, and *NonStop* TMF handles the locking.

Here's how locking works. When I go to read something so I can change it, if some user already has a lock on it, I have to wait. So I will sit there and pause in my program until the user releases that lock. Then I read the data in, and now I've got a lock on it. And when I start to update it, if some other user has read it, I have to wait for that user's transaction to complete before I can change it. So we have shared read locks, but all the update locks are private. I can't change a value and let someone else read it while my transaction is still alive. My transaction has to commit, all or nothing, so that my updates can become public.

Durability means that when you ask *NonStop* TMF to end the transaction, and the system replies OK, that means *durable* OK. It means if the whole world crashes in the next moment, it's still going to be OK, until someone begins a new transaction and changes the field that you just changed.

CHARLES JOHNSON, NonStop system architect

Charles Johnson intended to major in biology when he entered the University of San Francisco. Then chemistry. Then physics. But when he came face to face with a DEC PDP 11/34, he realized that computers were his future and left school to consult in the growing information technology industry. Among other things, his work took him to Israel for a year—where the Israeli Air Force was installing NonStop™ systems to keep track of F15/F16 fighter maintenance—and to Cape Canaveral, where he spent two years on space-station software. He accepted a job with Tandem (now Compaq's NonStop Division) in 1986 and joined the TMF group in 1989, where he continues to apply his system architecture and database expertise to the key TMF product. Johnson's work has resulted in 4 U.S. patents, with 10 more in the pipeline.

Q: You used the term *log serialization*. What's that all about?

A: Log serialization in databases is critical when there are multiple users. There may be thousands of transactions going on at the same time, and they're all touching the same data. How can that be, when everyone has an individual, isolated view of the world? Whenever I touch something, nobody else gets to play with it until I end the transaction. I have an isolated view of the database that allows me to make updates, and when I'm done, everything is either yes or no.

Q: You keep talking about databases. What else does *NonStop* TMF do?

A: It's true that *NonStop* TMF and databases are inextricably linked. *NonStop* TMF software is wrapped around DP2, the database resource manager for *NonStop Himalaya* systems, and DP2 is wrapped around *NonStop* TMF. But *NonStop* TMF also talks to the file system, to all the applications, to the operating system, and to network management, and it deals with load balancing. *NonStop* TMF is an integral part of the operating system of the *NonStop Himalaya* server, and it's the interface to most everything else in the system as well.

Q: If *NonStop* TMF is integral to the operating system, why write a separate article about it?

A: Because *NonStop* TMF is one of the most significant differentiators of *NonStop Himalaya* servers. Many *NonStop* system fundamentals—such as availability, linear scalability, and extreme transaction processing performance—are greatly enhanced by *NonStop* TMF. Some customers still aren't using *NonStop* TMF because 15 years ago TMF transaction

protection may have caused performance degradations in certain situations. That is no longer the case, and that's why we need this article.

Some customers use Enscribe files, for example, and they write through to disk instead of using transactions. By doing this, they incur very high response overhead. Many people don't understand that using transactions is much faster than not using them. Writing things through to disk takes a fraction of a second—so as your database gets bigger, the system becomes increasingly less scalable.

But with *NonStop* TMF, the disk process can change a cache block, and it doesn't get written to disk until maybe five minutes later. That's perfectly okay, it's perfectly safe, because the transaction manager has the update in the log, a safe place. System performance can be many, many times faster using transactions.

Q: What does it mean to write something through to disk?

A: It means you stop everything until the data has landed on the disk, the write has completed, and you get the reply. The interesting thing is that disk managers under *NonStop* TMF never have a consistent picture of their disks, because they don't do all of those random writes to make sure all the fields are consistent on the disk. They don't have to, because it's in the log. When data is written to the log in a serialized fashion, it doesn't have to be written to the data disk immediately. When you write to a log, you position the disk head on a track and just start writing, without moving the disk head back and forth. There's no access time change between writes, so it's very fast.

Q: Why do some *NonStop Himalaya* system users choose not to use *NonStop* TMF?

A: Bad memories. When TMF1 first came out in the early 1980s, it was the only thing in the world that could do what it did—but it wasn't terribly robust, it was extremely complex, and it detracted from system performance. The current version of the product doesn't have these problems, of course.

Q: Is *NonStop* TMF use increasing among *NonStop Himalaya* system users?

A: Yes, largely due to a product called *NonStop* AutoTMF, developed by Carr Scott Software and now a Compaq product. *NonStop* AutoTMF software can take applications that were not coded for *NonStop* TMF and make them use transactions and *NonStop* TMF automatically, without any special coding. So the system runs faster, without recoding. It's an amazing piece of software.

> "*NonStop* TMF is at the heart of the system, and at the heart of every other system that's connected to it. It guarantees that the truth of the database across the entire computing space is always consistent."

With *NonStop* AutoTMF, you're basically going from nontransactional database work—write this, write that, lock this, explicitly lock this, change the field, and then release the lock—to where you begin a transaction and then just do updates. You don't have to lock anything when you use *NonStop* TMF, because it acquires all the locks for you under the transaction. And then when you do ENDTRANSACTION, all those locks get released.

You never have to lock things, remember that they're locked, or deal with locking problems. *NonStop* TMF gets rid of that kind of code in people's programs.

Q: What kinds of enhancements are you making to *NonStop* TMF?

A: Most of our developments and enhancements relate to improving performance and availability. We also work closely with the Compaq Remote Database Facility (RDF) group. A lot of RDF software upgrades have come out in the last year—things like networked RDF and lockstep—and we've participated in those development efforts. RDF depends on *NonStop* TMF because it reads the log, gets all the changes to the database, and propagates them over to another node for disaster recovery.

Q: How does *NonStop* TMF tie in with the NonStop Division's NonStop Business initiative?

A: The NonStop Business initiative, formerly the Indestructible Scalable Computing initiative, depends on changes to a lot of areas to make

> "If you're not using *NonStop* TMF, you should be. *NonStop* TMF is all about the truth, the whole truth, and nothing but the truth."

things more available. We want to get availability to a point where the system is always there, where we can bring applications down and install new releases of software without losing any availability. Both *NonStop* TMF and DP2 are closely involved in making that happen. *NonStop* Business is in the design stage now.

Q: Are you doing anything to *NonStop* TMF specifically to make it work better in Compaq's Zero Latency Enterprise (ZLE) environment?

A: Yes, we continue to improve scalability and performance to complement ZLE environments. Network scalability is also important, so that the local transaction rate and the network transaction rate are the same. We're not quite there—the network transaction rate maximum is around 65 to 70 percent of the local *NonStop* TMF transaction rate. That's because networked *NonStop* TMF has to do two-phase commit between nodes to coordinate transaction commit and to guarantee serializability of the logs. But we're moving toward network transparency, and that will be important to ZLE environments in the future.

In the latest version of the ZLE e-CRM demo, for example, the *NonStop* system is performing 70,000 database calls per second, and the *NonStop* TMF subsystem has not demonstrated any bottlenecks. It's not even breathing hard, and it still has plenty of cycles for growth.

Q: Is *NonStop* TMF open?

A: Absolutely. And because it's open, we can deal with databases on other kinds of systems, such as Oracle, IBM DB2, and Microsoft SQL Server. Open *NonStop* TMF allows us to share transactions. We can import transactions from other systems, including Java transaction systems. *NonStop* TMF is completely interoperable with all these environments. And that means that if the transaction fails anywhere, all these databases get set back to their initial state, all their locks get released, they all get cleaned up. Without a transaction-based system, you cannot do that. You would never, ever get it right.

As I said, we can import transactions that other systems own. But the really great thing about *NonStop* TMF is nonblocking commit coordination. That means that the *NonStop Himalaya* server and its applications are always available. So if the *NonStop Himalaya* node is the parent, and everything else—IBM, Oracle, SQL Server—is the child, then they *always* get the answer as to whether the transaction was committed or aborted.

If I were out there in the world trying to do transactions, I'd always make sure I began them on a *NonStop Himalaya* server, no matter where they went from there. That way, the source of the transaction is always there to find out whether the transaction commits or aborts. If any node goes down, it can get its transactions resolved when it comes back up. Because it all started on a *NonStop Himalaya* server, everything is as available as it can be.

Q: What aspect of *NonStop* TMF are you proudest of?

A: That aspect of absolute truth, of always providing the ultimately true answer. We have a major customer in the financial services arena that handles corporate mergers and regularly runs transactions on *NonStop* TMF. This customer has done single transactions with a value of US$100 billion—that's a *single NonStop* TMF transaction. There is absolutely no other system in the known computing universe that anyone would trust with that kind of money. Other vendors have problems with system availability or database consistency, but you can always count on the Compaq *NonStop Himalaya* system.

Q: What's the main message you'd like to convey about *NonStop* TMF?

A: If you're not using *NonStop* TMF, you should be. *NonStop* TMF is all about the truth, the whole truth, and nothing but the truth. It's the magic behind the fundamental attributes of the *NonStop Himalaya* system: data integrity, reliability, parallelism, transparency, linear scalability, availability, and database consistency.

NonStop TMF is a complex, powerful product that can deliver huge performance gains for customers that use it. And we're making it even better, as we move toward true indestructible scalable computing in the *NonStop* system environment. ■