

# a conversation with NonStop TMF architect Charles Johnson

article

Charles Johnson practically cut his teeth on Transaction Monitoring Facility (TMF) software for HP 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 says wistfully, but with a touch of pride. The product was renamed NonStop Transaction Manager/MP in the 1990s. In 2002 the TMF name came back as NonStop Transaction Management Facility (NonStop TMF), but it was always TMF to Charles Johnson.

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

## What is TMF?

TMF is the facility that manages the interrelationship between all the fundamental elements of a NonStop 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 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 guarantee 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.

the magic behind  
hp NonStop systems

## So TMF plays a big part in ensuring database consistency?

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 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 happen, but a transaction never partially happens. With TMF, if your process fails halfway through a transaction, the transaction is aborted, and the database returns to the state at the beginning of that transaction. You don't have to remember a thing, because TMF remembers all of it. Without TMF, 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 TMF with the two-phase commit protocol for network commitment of transactions.

The nice thing about 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 TMF transaction protection, it all goes back to the initial state.

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

Not really. In TMF, if you don't talk about deep issues, you don't get to talk about much. The good news is that TMF will handle the most technical, complex details of transaction processing, so you and your programmers don't have to. 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 TMF much.

With 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.

### **OK then, can we get back to ACID?**

We 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 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 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.

### **You used the term "log serialization." What's that all about?**

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 his own 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.

### **You keep talking about databases. What else does TMF do?**

It's true that TMF and databases are inextricably linked. TMF is wrapped around DP2, the database resource manager for NonStop systems, and DP2 is wrapped around TMF. But 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. TMF is an integral part of the NonStop Kernel operating system, and it's the interface to most everything in the NonStop system.

### **If TMF is integral to the NonStop Kernel operating system, why write an article about it?**

Because TMF is one of the most significant differentiators of NonStop servers. Many NonStop fundamentals, such as availability, linear scalability, and extreme transaction processing performance, are greatly enhanced by TMF. Some customers still aren't using 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 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 OK, 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.

### **What does it mean to write something through to disk?**

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 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.

### **Why do some NonStop system users choose not to use TMF?**

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.

### **Is TMF use increasing among NonStop users?**

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

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 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. TMF gets rid of that kind of code in people's programs.

### **What kind of enhancements are you making to TMF?**

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

### **How does TMF tie in with the NonStop Enterprise Division's Indestructible Scalable Computing initiative?**

Indestructible scalable computing (ISC) depends on changes to a lot of areas to make 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 TMF and DP2 are closely involved in making that happen. ISC is in the design stage now.

### **Are you doing anything to TMF specifically to make it work better in the ZLE environment?**

Yes, we continue to improve scalability and performance to complement HP Zero Latency Enterprise (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 TMF transaction rate. That's because networked 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 TMF subsystem has not demonstrated any bottlenecks. It's not even breathing hard, and it still has plenty of cycles for growth.

## Is TMF open?

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 TMF allows us to share transactions. We can import transactions from other systems, including Java™ and Tuxedo transaction systems. 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 TMF is nonblocking commit coordination. That means that the NonStop server and its applications are always available. So if the NonStop node is the parent, and everything else—IBM, Oracle, SQL Server—is the child, then they *always* gets 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 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 server, everything is as available as it can be.

## What aspect of TMF are you proudest of?

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 TMF. This customer has done single transactions with a value of US\$100 billion—that's a *single* 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 NonStop system.

## What's the main message you'd like to convey about TMF?

If you're not using TMF, you should be. 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 system: data integrity, reliability, parallelism, transparency, linear scalability, availability, and database consistency.

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.



For more information, go to [www.hp.com/go/nonstopcontinuity](http://www.hp.com/go/nonstopcontinuity).

October 2002, first published October 2001. All product names mentioned herein may be trademarks of their respective companies. Java is a U.S. trademark of Sun Microsystems, Inc. Microsoft is a U.S. registered trademark of Microsoft Corporation. Oracle is a registered U.S. trademark of Oracle Corporation, Redwood City, California. HP shall not be liable for technical or editorial errors or omissions contained herein. The information is subject to change without notice. The warranties for HP products are set forth in the express limited warranty statements accompanying such products. Nothing herein should be construed as constituting an additional warranty.

5981-3869EN

©2002 Hewlett-Packard Company