

Software Transactional Memory

Ralf Westphal, ralfw@ralfw.de, <http://www.ralfw.de>

Freier Autor, Berater, Entwickler und Trainer

III PROFESSIONAL DEVELOPER COLLEGE
GRUNDLEGENDE ANGEWANDT INTEGRIERT
www.prodevcollege.de



Concurrency is the next major revolution in how we write software


Herb Sutter, Dr. Dobb's Journal, 30(3), March 2005

<http://www.gotw.ca/publications/concurrency-ddj.htm>

Concurrency Concerns

- Distribution of work
 - ThreadPool.QueueUserWorkItem(), OpenMP, Active C#
- Coordination of actions
 - WaitHandle, queues, Ports of CCR, protocols in Active C#
- Synchronization of shared resource access
 - Locks, Software Transactional Memory (STM), Space Based Collaboration (SBC)

Locking Challenges

- „Lock Leaks“
 - „What is opened must be closed“
- Inappropriate granularity
 - Balance maintainability with efficiency
- Deadlocks
 - It's all about order
- Resource inconsistencies 
 - Locks don't help with errors

Inconsistencies Despite Locks

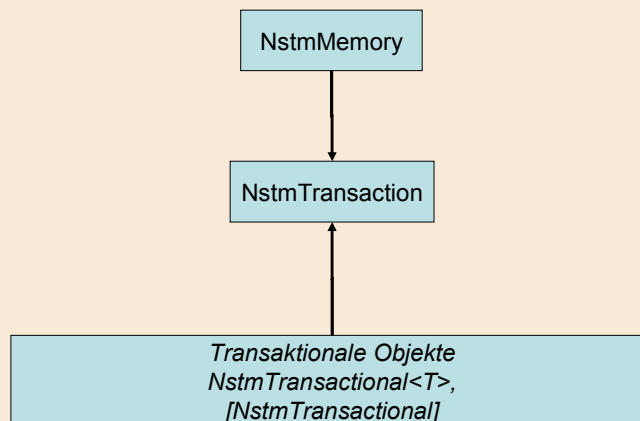
```
1 Account a, b;  
2 ...  
3 TransferMoney(a, b, 150);  
4 ...  
5 void TransferMoney(  
6     Account source,  
7     Account destination,  
8     double amount)  
9 {  
10     Account a1 = source;  
11     Account a2 = destination;  
12     if (a2.LockIndex < a1.LockIndex)  
13     {  
14         Account t;  
15         t = a2;  
16         a2 = a1;  
17         a1 = t;  
18     }  
19  
20     lock(a1)  
21     lock(a2)  
22     {  
23         destination.Balance += amount;  
24         source.Balance -= amount;  
25     }  
26 }
```

```
100 class Account  
101 {  
102     private double balance;  
103     ...  
104     public double Balance  
105     {  
106         get { return this.balance; }  
107         set  
108         {  
109             if (value < 0)  
110                 throw new ApplicationException(...);  
111             this.balance = value;  
112         }  
113     }  
114 }
```



.NET STM – Architecture Overview

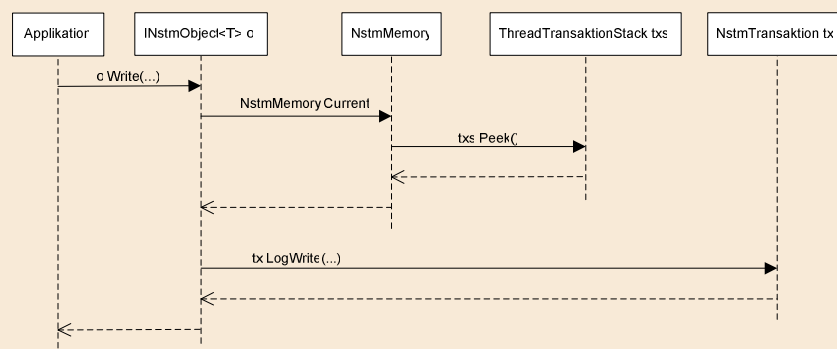
Open Source @ <http://code.google.com/p/nstm/>



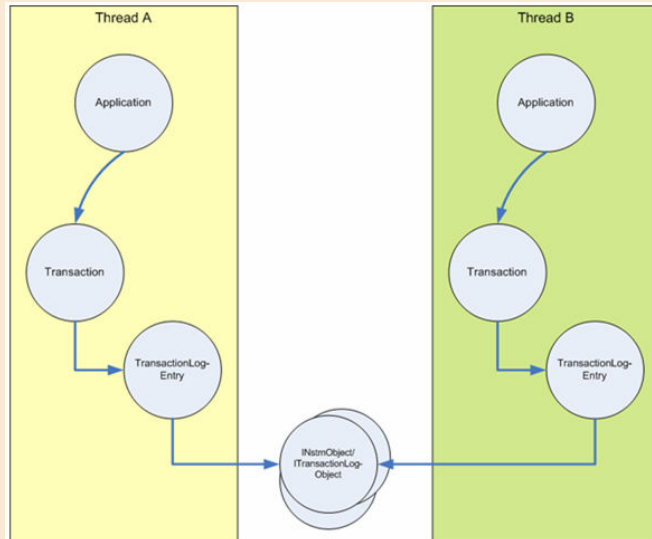
NSTM „Hello, World“ v1.0

```
1 static void Main()
2 {
3     INstmObject<string> greeting;
4     greeting = NstmMemory.CreateObject<string>();
5     greeting.Write("hello, world!");
6
7     using (INstmTransaction tx = NstmMemory.BeginTransaction())
8     {
9         greeting.Write("bye, bye!");
10        tx.Rollback();
11    }
12
13    Console.WriteLine(greeting.Read());
14 }
```

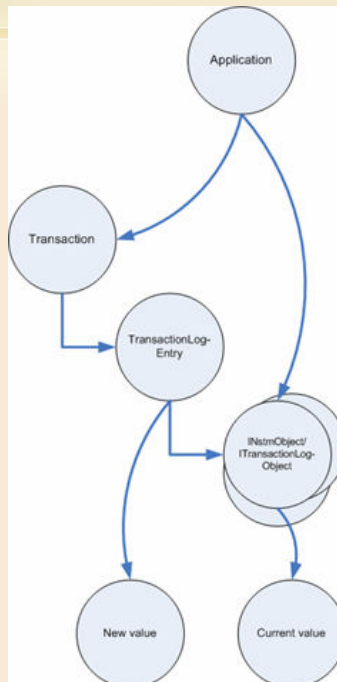
Inside NSTM



Thread Afinity



Isolation



NSTM „Hello, World“ v2.0

```
1 static void Main()
2 {
3     NstmTransactional<string> greeting;
4     greeting = "hello, world!";
5
6     using (INstmTransaction tx = NstmMemory.BeginTransaction())
7     {
8         greeting.Value = "bye, bye!";
9         tx.Rollback();
10    }
11
12    Console.WriteLine(greeting.Value);
13 }
```

Transactional Objects v1.0

```
1 static void Main()
2 {
3     INstmObject<Person> txp;
4     txp = NstmMemory.CreateObject<Person>(
5         new Person("john", new DateTime(1972, 4, 27)));
6
7     using (INstmTransaction tx = NstmMemory.BeginTransaction())
8     {
9         Person p = txp.Read();
10        p.name = "paul";
11        txp.Write(p);
12
13        tx.Commit();
14
15        Console.WriteLine(txp.Read());
16 }

```

```
class Person : ICloneable
{
    public string name;
    public DateTime dob;

    public Person(string name, DateTime dob)
    {
        this.name = name;
        this.dob = dob;
    }

    public override string ToString()
    {
        return name + ", " + dob.ToString();
    }
}

```

ICloneable Members

Transactional Objects v2.0

```

1 static void Main()
2 {
3     PersonTx p = new PersonTx("john", new DateTime(1972, 4, 27));
4
5     using (INstmTransaction tx = NstmMemory.BeginTransaction())
6     {
7         p.name = "paul";
8         tx.Commit();
9     }
10
11    Console.WriteLine(p);
12 }

```

```

[NstmTransactional]
class PersonTx
{
    public string name;
    public DateTime dob;

    public PersonTx(string name, DateTime dob)
    {
        this.name = name;
        this.dob = dob;
    }

    public override string ToString()
    {
        return name + ", " + dob.ToString();
    }
}

```

STM for Consistency

```

1 static void Main()
2 {
3     AccountTx myAccount = new AccountTx(); myAccount.amount = 0;
4     AccountTx yourAccount = new AccountTx(); yourAccount.amount = 1000;
5
6     INstmTransaction tx = NstmMemory.BeginTransaction();
7     try
8     {
9         Transfer(myAccount, yourAccount, 350);
10        tx.Commit();
11    }
12    catch (ApplicationException ex)
13    { Console.WriteLine("*** Error during money transfer: {0}", ex.Message); }
14    finally
15    { tx.Rollback(); }
16
17    Console.WriteLine("my: {0}, yours {1}", myAccount.amount, yourAccount.amount);
18 }
19
20 static void Transfer(AccountTx destination, AccountTx source, int amount)
21 {
22     source.amount -= amount;
23     throw new ApplicationException("argh!");
24     destination.amount += amount;
25 }

```

Guess what...

```

1 static void Main()
2 {
3     PersonTx2 p = new PersonTx2("john", new DateTime(1972,5,12));
4     p.addr = new AddressTx("london");
5
6     using (INstmTransaction txOut =
7         NstmMemory.BeginTransaction())
8     {
9         p.name = "paul";
10
11        using(INstmTransaction txIn =
12            NstmMemory.BeginTransaction())
13        {
14            p.addr.city = "new york";
15            txIn.Commit();
16        }
17
18        txOut.Rollback();
19
20    Console.WriteLine(p);
21 }

```

```

[NstmTransactional]
class PersonTx2
{
    public string name;
    public DateTime dob;
    public AddressTx addr;

    public PersonTx2(string name,
    {
        this.name = name;
        this.dob = dob;
    }
    |
    public override string ToString()
    {
        return string.Format("Per
    }
}

[NstmTransactional]
class AddressTx
{
    public string city;

    public AddressTx(string city)
    {
        this.city = city;
    }
}

```

```

C:\WINDOWS\system32\cmd.exe
Person(john, 12.05.1972, Address(london))
Drücken Sie eine beliebige Taste . . .

```

Transaction Scopes

```

static void Main()
{
    PersonTx2 p = new PersonTx2("john", new DateTime(1972,5,12));
    p.addr = new AddressTx("london");

    using (INstmTransaction txOut = NstmMemory.BeginTransaction())
    {
        p.name = "paul";

        using (INstmTransaction txIn =
            NstmMemory.BeginTransaction(NstmTransactionScopeOption.Required,
            NstmTransactionScopeOption.RequiresNested,
            NstmTransactionScopeOption.RequiresNew,
            NstmTransactionScopeOption.RequiresNew))
        {
            p.addr.city = "new york";
            txIn.Commit();
        }

        txOut.Rollback();
    }

    Console.WriteLine(p);
}

```

```

C:\WINDOWS\system32\cmd.exe
Person(paul, 12.05.1972, Address(new york))
Drücken Sie eine beliebige Taste . . .
C:\WINDOWS\system32\cmd.exe
Person(john, 12.05.1972, Address(new york))
Drücken Sie eine beliebige Taste . . .

```

Transaction Validation

When	Condition (isolation level + clone mode)	What (read mode of txo)
Validate on read and on commit	serializable + cloneOnWrite	ReadOnly (on read), ReadOnly+ReadWrite (on commit)
Validate on commit only	serializable + cloneOnRead	ReadOnly+ReadWrite
Validate on commit	readCommitted + cloneOnRead	ReadOnly
no validation	readCommitted + cloneOnWrite	-

Implicit Transactions

- Integration with .NET System.Transactions

```

1 NstmMemory.SystemTransactionMode = NstmSystemTransactionMode.EnlistOnAccess;
2 NstmTransactional<int> iTx = ...;
3 ...
4 using (TransactionScope tx = new TransactionScope())
5 {
6     iTx.Value = ...;
7     tx.Complete();
8 }

```

Heterogeneous Transactions

```

1 const int amountToTransfer = 350;
2
3 AccountTx myAccount = new AccountTx(); myAccount.amount = 0;
4 AccountTx yourAccount = new AccountTx(); yourAccount.amount = 1000;
5
6 NstmMemory.SystemTransactionsMode = NstmSystemTransactionsMode.EnlistOnAccess;
7 using (TransactionScope tx = new TransactionScope())
8 {
9     yourAccount.amount -= amountToTransfer;
10    myAccount.amount += amountToTransfer;
11
12    using (SqlConnection conn = new SqlConnection(@"..."))
13    {
14        conn.Open();
15        SqlCommand cmd = new SqlCommand("insert into messages (msg) values (@message)",
16        conn);
17        cmd.Parameters.AddWithValue(
18            "@message",
19            string.Format("tranfered {0} @ {1}", amountToTransfer, DateTime.Now));
20        cmd.ExecuteNonQuery();
21    }
22    tx.Complete();
23 }

```

Declarative Transactions

- In-line

```

1 NstmMemory.ExecuteAtomically(
2     delegate
3     {
4         ...
5     }
6 );

```

Can be retried!

- Methods

```

21 [NstmAtomic()]
22 static void TransferMoney(int amountToTransfer)
23 {
24     myAccount.Write(myAccount.Read() - amountToTransfer);
25     yourAccount.Write(yourAccount.Read() + amountToTransfer);
26     throw new ApplicationException("Money transaction failed!");
27 }

```

Transactional Collections

All collections need to be rewritten using (N)STM transactional objects

```
1 [NstmTransactional]
2 public class NstmStack<T>
3 {
4     [NstmTransactional]
5     public class Entry
6     {
7         private T value;
8         private Entry next;
9     }
10    public Entry(T value)
11    {
12        this.value = value;
13    }
14    public T Value
15    {
16        {
17            get { return this.value; }
18            set { this.value = value; }
19        }
20    }
21    public Entry Next
22    {
23        {
24            get { return this.next; }
25            set { this.next = value; }
26        }
27    }
28    private Entry top;
29    private int count;
30    public NstmStack()
31    {
32        this.top = null;
33        this.count = 0;
34    }
35    public void Push(T value)
36    {
37        Entry newEntry = new
38        Entry(value);
39        newEntry.Next = this.top;
40        this.top = newEntry;
41        this.count++;
42    }
43 }
```

Summary

- STM
 - Established programming model
 - Intuitive access to resources
 - Solves problems with locks
 - No lock leaks
 - Automatic, fine granularity
 - No deadlocks
 - No inconsistencies
- NSTM
 - Managed Code
 - Open Source
 - Implements well-known data access pattern
 - Intuitive thru AOP
 - PostSharp Laos (www.postsharp.org)

STM makes access to shared resources
from concurrent code as easy as database
programming.

Questions?

Resources

- Wikipedia, Software transactional memory, http://en.wikipedia.org/wiki/Software_transactional_memory
- Tim Harris et al., Composable Memory Transactions, <http://research.microsoft.com/~simonpj/papers/stm/stm.pdf>
- Microsoft Research, C# Software Transactional Memory, <http://research.microsoft.com/research/downloads/Details/6cfc842d-1c16-4739-afaf-edb35f544384/Details.aspx>
- Ralf Westphal, Software Transactional Memory, <http://weblogs.asp.net/ralfw/archive/tags/Software+Transactional+Memory/default.aspx>
- Ralf Westphal, NSTM Implementation, <http://www.codeplex.com/NetSTM>
- Ralf Westphal, Datenkonsistenz beim Multithreading sichern, dotnetpro 9/07

Über den Referenten

- Ralf Westphal (www.ralfw.de) ist freier Softwaretechnologievermittler. Er arbeitet als Fachautor mit mehr als 300 Publikationen, Coach/Berater und Referent auf Entwickler-Events im In- und Ausland.
- Schwerpunkt seiner Arbeit sind die Architektur von .NET-Software und die Förderung innovativer Softwaretechnologien. Bei der Wissensvermittlung beschreitet er gerne ungewöhnliche Wege, so zum Beispiel mit den Videoserien .NET TV und dotnetpro.tv und dem Trainingsunternehmen Professional Developer College (www.prodevcollege.de).
- Ralf Westphal ist Microsoft „Visual Developer Solution Architect“ MVP und war von 1998 bis 2005 einer der unabhängigen Microsoft Regional Directors für Deutschland.
- Email: ralfw@ralfw.de



III PROFESSIONAL DEVELOPER COLLEGE
THE EXCITING WAY TO MORE .NET COMPETENCE
www.prodevcollege.de

Publications

Bücher



.NET kompakt, Spektrum Akademischer Verlag 2002, ISBN 3827411858



ADO.NET Datenbankprogrammierung, Addison-Wesley 2002, ISBN 3827319978



Jetzt lerne ich ADO.NET, Markt+Technik, 2003, ISBN 3827262291
(zusammen mit Christian Weyer)



.NET 3.0 kompakt, Spektrum Akademischer Verlag 2007, ISBN 382741458X
(zusammen mit Christian Weyer)

In Fachzeitschriften



Video



dotnetpro.tv

www.dotnettv.de

tv.dotnetpro.de