

Operation and Implementation of Composite Flowops in FileBench

Andrew W. Wilson
Sun Microsystems
©2008 Sun Microsystems, inc.

Introduction

Most computer programming languages have some notion of subroutines, which allow a common chunk of code to be used by different sections of the program. The FileBench workload modeling language currently does not have such a feature, which can result in having to duplicate many flowops if operations with varied percentages are required. If such a feature were defined, significant run time savings, as well as more compact and understandable workload specifications would result.

An example where a subroutine like feature would be useful is the [webserver.f](#) workload. It consists of ten triplets of `openfile`, `readwholefile`, `closefile` flowops, then one `appendfile` flowop, for a total of thirty one flowops. A fragment of the flowop definition portion of the model is shown in [Figure 1](#).

```
flowop openfile name=openfile1,filesetname=bigfileset,fd=1
flowop readwholefile name=readfile1,fd=1
flowop closefile name=closefile1,fd=1

*** eight more triplets ***

flowop openfile name=openfile10,filesetname=bigfileset,fd=1
flowop readwholefile name=readfile10,fd=1
flowop closefile name=closefile10,fd=1
flowop appendfilerand name=appendlog,filesetname=logfiles,iosize=16k,fd=2
```

Figure 1: Flowops from webserver.f workload model.

If those three flowops were combined into one aggregate flowop, the thirty one total flowops would be reduced to eleven. Actually, the proposed method does even better than that.

Composite Flowop Description

The new addition to FileBench's workload modeling language is the composite flowop. A composite flowop is a list of flowops which are invoked each time the composite flowop is invoked. A key advantage of this approach is that composite flowops can have iteration counts supplied when they are invoked, so a sequence of flowops that needs to be repeated several times for each pass through the overall list of flowops can be placed in a composite flowop and then given an appropriate iteration count. Using a single composite flowop, the `webserver.f` workload can be recast as shown in [websrv.f](#) which features a total of only five flowops, including the workload defined composite flowop. The

main fragment of the workload model now becomes that shown in [Figure 2](#).

```

define flowop name=readone
{
    flowop openfile name=openfile1,filesetname=bigfileset,fd=1
    flowop readwholefile name=readfile1,fd=1
    flowop closefile name=closefile1,fd=1
}

define process name=filereader,instances=1
{
    thread name=filereaderthread,memsize=10m,instances=$nthreads
    {
        flowop readone name=work, iters=10
        flowop appendfilerand name=appendlog,filesetname=logfiles,iosize=16k,fd=2
    }
}

```

Figure 2: Flowop and process fragment of webserv.f workload model

The above example illustrates the use of composite flowops to coalesce a set of identical operations into one which is then executed multiple times. Composite flowops can be made even more useful by allowing parameters to be passed into them at the time of their invocation. This is done by creating another new concept for the workload modeling language of local variables. These are variables which are created within the context of a composite flowop, and can be referenced by flowops within it. The local variables are initially specified as part of the composite flowop definition, and later set to appropriate values when the flowop is invoked. If we wanted to read from two different filesets with the webserv.f workload, we could use a local variable to achieve this, as illustrated in [Figure 3](#).

```

define flowop name=readone, $fsname
{
    flowop openfile name=openfile1,filesetname=$fsname,fd=1
    flowop readwholefile name=readfile1,fd=1
    flowop closefile name=closefile1,fd=1
}

define process name=filereader,instances=1
{
    thread name=filereaderthread,memsize=10m,instances=$nthreads
    {
        flowop readone name=work, iters=5, $fsname=bigfileset1
        flowop readone name=work, iters=5, $fsname=bigfileset2
        flowop appendfilerand name=appendlog,filesetname=logfiles,iosize=16k,fd=2
    }
}

```

Figure 3: Fragment of modified web server workload; webserv.f

Unlike subroutines as used in many traditional programming languages, such as “C”, the local variables are explicitly named in the invocation, so that there is no position dependence on variable assignment. Note that assignment can also be done on the composite flowop definition statement, which will essentially provide a default value for the local variable. Both strings and integers can be passed through local variables.

Composite flowop definitions can include invocations of previously defined composite flowops. [Figure 4](#) shows a fragment of a possible fileserver workload model which demonstrates composite flowop nesting, [efsw.f](#).

```
define flowop name=readwrite, $fileset
{
  flowop openfile name=openfile4, filesetname=$fileset, fd=1
  flowop openfile name=openfile5, filesetname=$fileset, fd=2
  flowop readwholefile name=readfile1, fd=1
  flowop writewholefile name=writefile1, fd=2, srcfd=1
  flowop closefile name=closefile4, fd=1
  flowop closefile name=closefile5, fd=2
}

define flowop name=dowork, $filesetnm, $rwriters
{
  flowop createfile name=createfile1, filesetname=$filesetnm, fd=1
  flowop appendfilerand name=appendfilerand1, iosize=$meaniosize, fd=1
  flowop closefile name=closefile1, fd=1
  flowop readwrite name=rw1, iters=$rwriters, $fileset=$filesetnm
  flowop deletefile name=deletefile1, filesetname=$filesetnm
  flowop statfile name=statfile1, filesetname=$filesetnm
}

define process name=filereader1, instances=1
{
  thread name=user1, memsize=10m, instances=$nthreads
  {
    flowop dowork name=dowork1, iters=1, $rwriters=5, $filesetnm=user1fileset
  }
  *** more threads ***
}
```

Figure 4: Nested composite flowop example.

In Figure 4 there is a composite flowop defined with a type name of *readwrite*, and a second with a type name of *dowork*. The *readwrite* composite flowop has a local variable named *\$fileset* which is used to open a pair of the fileset's files for some I/O operations. The *dowork* composite flowop invokes the *readwrite* composite flowop with an iteration count supplied by the local variable *\$rwriters*, and assigns *readwrite*'s *\$fileset* local variable to its *\$filesetnm* local variable. The *\$filesetnm* variable is also used to pass the fileset's name to several other flowops in *dowork*.

This workload model has multiple filesets each accessed by *\$nthreads* per fileset. As shown, thread *user1* access fileset *user1fileset*, and additional threads and filesets might be similarly named, though the exact names don't matter as long as they are unique. The *dowork* flowop is invoked by each of these threads which pass their specific values for *readwrite* flowop iterations to *\$rwriters* and *fileset*

name to \$filesetnm as part of their invocations of dowork. The values passed to \$rwriters and \$filesetnm can be fixed, as shown here, or can be obtained from global variables.

[Figure 5](#) shows the syntax for composite flowops. Each newly defined composite flowop must have a unique type name declared using the *name* attribute. It can optionally have a default iteration count supplied and any local variables must be declared, with optional default values. The optional local variable declarations are followed by a brace enclosed list of flowops that will be executed each time the composite flowop is invoked.

A composite flowop, once defined, can be invoked anywhere a normal flowop can be, and with a similar syntax, as shown in [Figure 5](#). However, as with the declaration, the only attribute allowed is the iteration count (*iters*). The last portion of the composite flowop invocation is a list of local variable assignments for some or all of the local variables declared as part of the composite flowop definition. These assignments can be to integer or string values, or local or global variables.

Defining a Composite Flowop:

```
define flowop name = <type name> [, iters=<default iteration count>] [, $<local
variable>[=<default value | global variable>]] [, more local variable declarations]
{
  flowop <type name> name=<name> [, <attributes> ...]
  [ ... ]
}
```

Invoking a Composite Flowop:

```
flowop <type name> name=<name> [, iters=<iteration count>] [, $<local
variable>=<value | variable>] [, more local variable assignments ...]
```

Figure 5: Composite Flowop Syntax

As you can see, very powerful models can be built with this approach. Multiple composite flowops can be defined, and they can be nested. It is necessary to define a composite flowop before invoking it, so any composite flowops which invoke other composite flowops must follow them in the workload file, and the composite flowops must also be specified before any threads which use them.

Appendices

Webserver.f

Listing of webserver.f:

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2007 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)webserver.f          1.1      07/05/31 SMI"

set $dir=/tmp
set $nfiles=1000
set $dirwidth=20
set $filesize=16k
set $nthreads=100

define fileset
name=bigfileset,path=$dir,size=$filesize,entries=$nfiles,dirwidth=$dirwidth,prealloc=100
define fileset
name=logfiles,path=$dir,size=$filesize,entries=1,dirwidth=$dirwidth,prealloc

define process name=filereader,instances=1
{
  thread name=filereaderthread,memsize=10m,instances=$nthreads
  {
    flowop openfile name=openfile1,filesetname=bigfileset,fd=1
    flowop readwholefile name=readfile1,fd=1
    flowop closefile name=closefile1,fd=1
    flowop openfile name=openfile2,filesetname=bigfileset,fd=1
    flowop readwholefile name=readfile2,fd=1
```

```

flowop closefile name=closefile2,fd=1
flowop openfile name=openfile3,filesetname=bigfileset,fd=1
flowop readwholefile name=readfile3,fd=1
flowop closefile name=closefile3,fd=1
flowop openfile name=openfile4,filesetname=bigfileset,fd=1
flowop readwholefile name=readfile4,fd=1
flowop closefile name=closefile4,fd=1
flowop openfile name=openfile5,filesetname=bigfileset,fd=1
flowop readwholefile name=readfile5,fd=1
flowop closefile name=closefile5,fd=1
flowop openfile name=openfile6,filesetname=bigfileset,fd=1
flowop readwholefile name=readfile6,fd=1
flowop closefile name=closefile6,fd=1
flowop openfile name=openfile7,filesetname=bigfileset,fd=1
flowop readwholefile name=readfile7,fd=1
flowop closefile name=closefile7,fd=1
flowop openfile name=openfile8,filesetname=bigfileset,fd=1
flowop readwholefile name=readfile8,fd=1
flowop closefile name=closefile8,fd=1
flowop openfile name=openfile9,filesetname=bigfileset,fd=1
flowop readwholefile name=readfile9,fd=1
flowop closefile name=closefile9,fd=1
flowop openfile name=openfile10,filesetname=bigfileset,fd=1
flowop readwholefile name=readfile10,fd=1
flowop closefile name=closefile10,fd=1
flowop appendfilerand name=appendlog,filesetname=logfiles,iosize=16k,fd=2
}
}

echo "Webserver Version 1.13 2005/06/21 21:18:53 personality successfully loaded"
usage "Usage: set \${dir}=<dir>"
usage "      set \${filesize}=<size>      defaults to \${filesize}"
usage "      set \${nfiles}=<value>      defaults to \${nfiles}"
usage "      set \${dirwidth}=<value>      defaults to \${dirwidth}"
usage "      set \${nthreads}=<value>      defaults to \${nthreads}"
usage "      run runtime (e.g. run 60)"

```

Listing of webserv.f:

```

#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.

```

```

# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2007 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)webserv.f          1.1      07/10/22 SMI"

set $dir=/export/home/tmp
set $nfiles=1000
set $dirwidth=20
set $filesize=16k
set $nthreads=10

define fileset
name=bigfileset,path=$dir,size=$filesize,entries=$nfiles,dirwidth=$dirwidth,preallo
c=100
define fileset
name=logfiles,path=$dir,size=$filesize,entries=1,dirwidth=$dirwidth,prealloc

define flowop name=readone
{
    flowop openfile name=openfile1,filesetname=bigfileset,fd=1
    flowop readwholefile name=readfile1,fd=1
    flowop closefile name=closefile1,fd=1
}

define process name=filereader,instances=1
{
    thread name=filereaderthread,memsize=10m,instances=$nthreads
    {
        flowop readone name=work, iters=10
        flowop appendfilerand name=appendlog,filesetname=logfiles,iosize=16k,fd=2
    }
}

echo "Webserver Version 1.13 2005/06/21 21:18:53 personality successfully loaded"
usage "Usage: set \$dir=<dir>"
usage "      set \$filesize=<size> defaults to $filesize"
usage "      set $nfiles=<value> defaults to $nfiles"
usage "      set $dirwidth=<value> defaults to $dirwidth"
usage "      set $nthreads=<value> defaults to $nthreads"
usage "      run runtime (e.g. run 60)"

```

Listing of efs.w.f:

```

#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2007 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)efsw.f      1.1      07/10/22 SMI"

set $dir=/tmp
set $nfiles=700
set $meandirwidth=20
set $filesize=128k
set $nthreads=10
set $meaniosize=16k

define fileset
name=usr1fileset,path=$dir,size=$filesize,entries=$nfiles,dirwidth=$meandirwidth,pr
ealloc=80, paralloc

define fileset
name=usr2fileset,path=$dir,size=$filesize,entries=$nfiles,dirwidth=$meandirwidth,pr
ealloc=80, paralloc

define fileset
name=usr3fileset,path=$dir,size=$filesize,entries=$nfiles,dirwidth=$meandirwidth,pr
ealloc=80, paralloc

define flowop name=readwrite, $fileset
{
  flowop openfile name=openfile4,filesetname=$fileset,fd=1
  flowop openfile name=openfile5,filesetname=$fileset,fd=2
  flowop readwholefile name=readfile1,fd=1
  flowop writewholefile name=writefile1,fd=2,srcfd=1
}

```

```

flowop closefile name=closefile4,fd=1
flowop closefile name=closefile5,fd=2
}

define flowop name=dowork, $filesetnm, $rwriters
{
  flowop createfile name=createfile1,filesetname=$filesetnm,fd=1
  flowop appendfilerand name=appendfilerand1,iosize=$meaniosize,fd=1
  flowop closefile name=closefile1,fd=1
  flowop readwrite name=rw1, iters=$rwriters, $fileset=$filesetnm
  flowop deletefile name=deletefile1,filesetname=$filesetnm
  flowop statfile name=statfile1,filesetname=$filesetnm
}

define process name=filereader1,instances=1
{
  thread name=usr1thread,memsize=10m,instances=$nthreads
  {
    flowop dowork name=dowork1, iters=1, $rwriters=5, $filesetnm=usr1fileset
  }

  thread name=usr2hread,memsize=10m,instances=$nthreads
  {
    flowop dowork name=dowork2, iters=1, $rwriters=4, $filesetnm=usr2fileset
  }

  thread name=usr3thread,memsize=10m,instances=$nthreads
  {
    flowop dowork name=dowork3, iters=1, $rwriters=3, $filesetnm=usr3fileset
  }
}

echo "efsw fileserver workload Version 1.00 2007/10/22 personality successfully
loaded"
usage "Usage: set \ $dir=<dir>"
usage "      set \ $filesize=<size>      defaults to $filesize"
usage "      set \ $nfiles=<value>      defaults to $nfiles"
usage "      set \ $nthreads=<value>      defaults to $nthreads"
usage "      set \ $meaniosize=<value> defaults to $meaniosize"
usage "      set \ $meandirwidth=<size> defaults to $meandirwidth"
usage "(sets mean dir width and dir depth is calculated as log (width, nfiles)"
usage " "
usage "      run runtime (e.g. run 60)"

```