DB-8: Highly Parallel Dump & Load

Tom Bascom
tom@greenfieldtech.com

PROGRESS SOFTWARE Exchange 2006

Agenda...

- **Why Dump and Load?**
- **What Methods are Available?**
- **Using a Highly Parallel Approach**
- **Demo!**

Why Dump and Load?

- **Because it is a X months since the last time.**
- **Because the vendor said so.**
- **Because we must in order to upgrade.**
- **Because it will improve performance.**

- **In order to move to a new platform.**
- **In order to reconfigure a storage management decision.**
- **In order to take advantage of a hot new db feature!**

Some Dump & Load Choices

- **“Classic” Dictionary Dump & Load**
- **Bulk Loader**
- **Binary Dump & Load**
- **Vendor Supplied Utilities**

- **Parallelization**
  - **Many Benefits**
  - **Difficult to Effectively Balance**
- **Automating the process**

Classic Dictionary D&L

1st the dump...

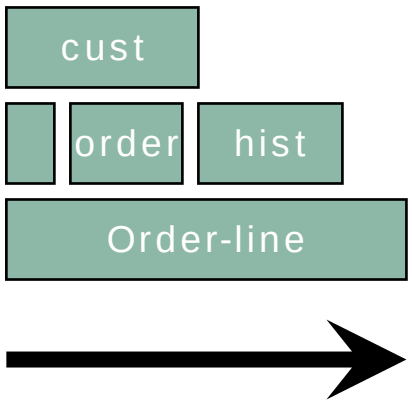
... and then the load.

Classic Dictionary D&L

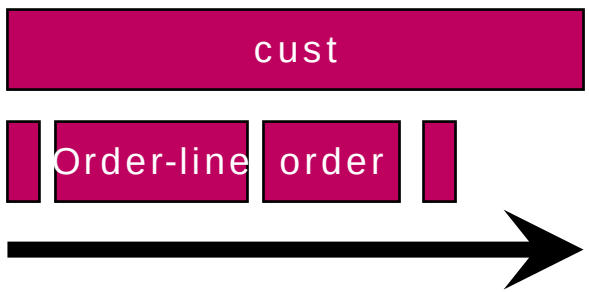
This can take DAYS (or even WEEKS!)

Multi-threaded D&L

1st the dump...



... and then the load.



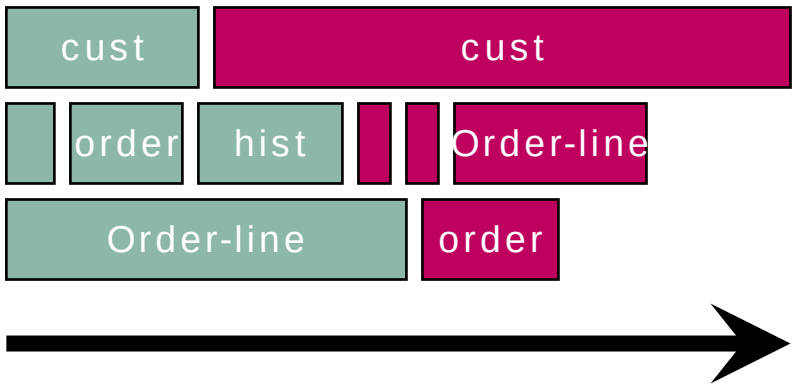
Multi-threaded D&L

- **Much more reasonable than “classic”**
- **Generally can be done in a weekend**

- **But still too long for many situations**

Parallel Balanced D&L

Both the dump...



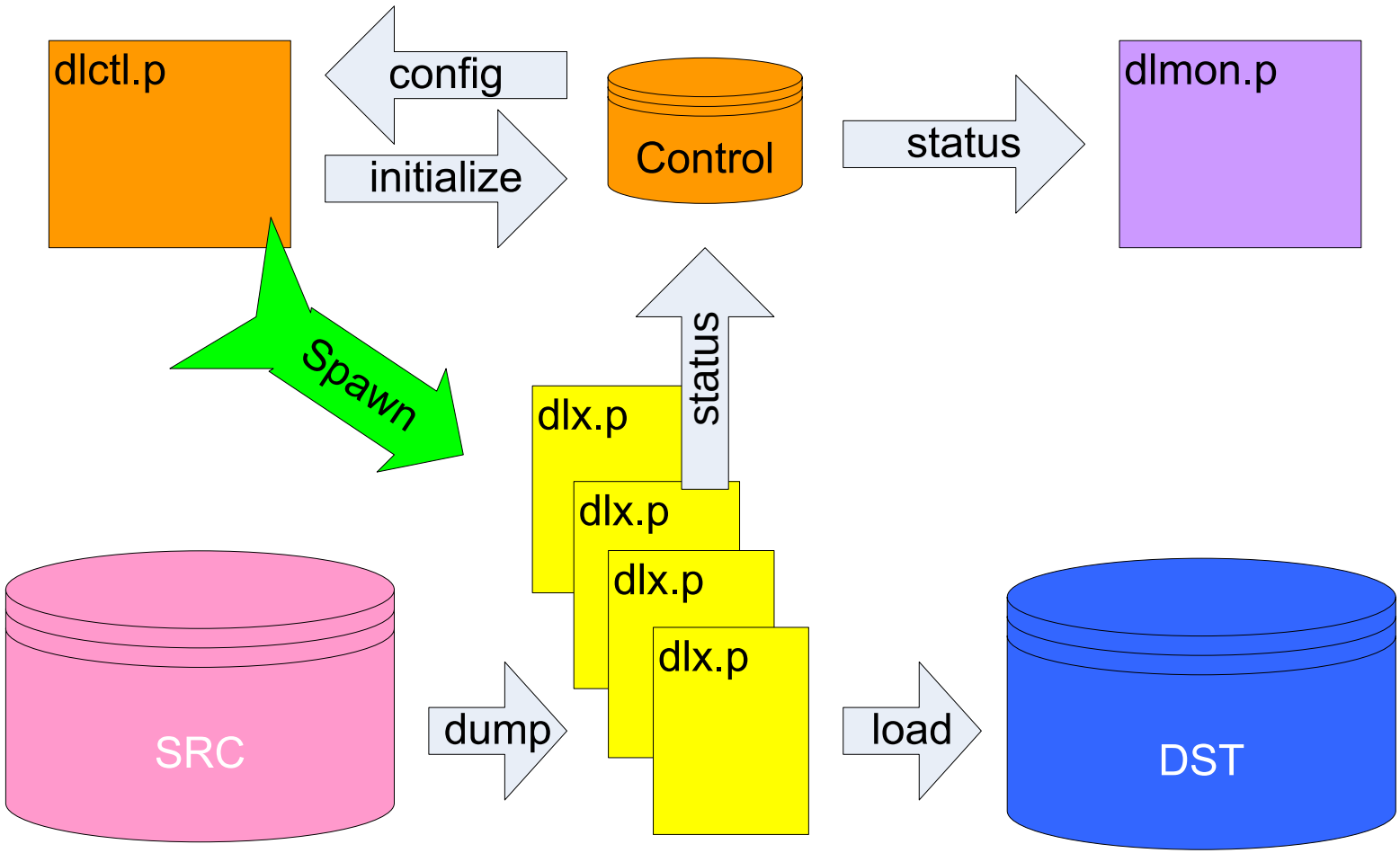
... and the load – in parallel, multi-threaded and load-balanced.

Parallel Balanced D&L

- **Usually a big improvement**
- **Often can be done in hours**

- **But difficult to configure and manage**

Highly Parallel D&L



Demo !!!

Resources

- **SUSE Linux 9 Server**
  - **2 x 2ghz (Xeon w/HT)**
  - **512 MB RAM**
  - **3 internal disks**
- **Various External Disks**
- **5GB Demo Database**
  - **85 Tables**
  - **250 Indexes**
  - **Various table and record sizes**

The Storage Area Design

- **Separate Tables from Indexes.**
- **Use a single Type 1 Area for tiny "miscellaneous" tables that have low activity.**
- **Isolate Small, but very active, tables in Type 2 Areas (or standalone areas if v9).**
- **Organize Large Tables by RPB in Type 2 Areas**
  - **Any table over 1 million rows**
  - **Any table over 1GB**
- **Isolate very large indexes**

The Storage Area Design

```
b exch06.b1   f   384000
#
d "Schema Area":6,64  .
#
d "Misc":10,256 exch06_10.d1   f   768
#
d "SmallTables":12,128;8 exch06_12.d1    f   9120
#
d "RPB256":14,256;512 exch06_14.d1    f   221280
#
d "RPB128":16,128;512 exch06_16.d1    f   768000
#
d "RPB64":18,64;512 exch06_18.d1    f   1536000
d "RPB64":18,64;512 exch06_18.d2    f   1536000
#
d "RPB32":20,32;512 exch06_20.d1    f   2048000
d "RPB32":20,32;512 exch06_20.d2    f   2048000
#
d "RPB16":22,16;512 exch06_22.d1    f   768000
#
d "Indexes":99,8 exch06_99.d1   f   1024000
```

Buffer-Copy & Raw-Transfer

- **Very Fast**
- **Eliminates The Middle Man (temp file IO operations)**
- **Provides fine-grained, 4GL level of control**
  - **Allows on the fly data manipulation**
  - **Useful when merging databases**
- **Can use with remote connections to bridge version numbers.**

- **In OpenEdge® 10.1 performance is essentially equal**
- **Cannot use RAW-TRANSFER with -RO**

Highly Parallel Dump & Load

- **90 threads.**
- **Dump threads are the same as load threads.**
- **Easy monitoring and control.**
- **No latency – the load finishes when the dump finishes.**
- **No intermediate disk IO:**
  - **More efficient usage of disk resources.**
  - **Higher overall throughput.**
- **No index rebuild.**
- **No need for table analysis.**

Dump Factors

- **Major factors that contribute to dump time:**
  - **Data distribution**
  - **-B, -Bp -RO**
  - **Storage Area Configuration**
  - **Disk Configuration and Throughput**

- **proutil dbname –C tabanalys (when doing a binary d&l)**

Take a Load Off

- **Factors that contribute to load time:**
  - **Progress version**
  - **Build indexes**
  - **-i, -bigrow, -B, -TB, -TM, -SG**
  - **Storage Area Configuration**
  - **Disk Configuration and Throughput**
  - **-T files**
  - **-noautoreslist (aka “forward-only”)**
- **The importance of testing**

The 80/20 Rule

- **80% of the tables are done very quickly**
- **20% of the tables take most of the time…**
- **… and nearly all of the space**

- **These are the tables that we target with multiple threads!**

Multiple Threads

```
Table    t Criteria
-------- - ------------------------------------------------------------------

document 1 where src.document.app#  < 1000000
document 2 where src.document.app# >= 1100000 and src.document.app# < 1200000
document 3 where src.document.app# >= 1200000 and src.document.app# < 1300000
document 4 where src.document.app# >= 1300000 and src.document.app# < 1400000
document 5 where src.document.app# >= 1400000 and src.document.app# < 1500000
document 6 where src.document.app# >= 1500000 and src.document.app# < 1600000
document 7 where src.document.app# >= 1600000

LOGS     1 where type = "P" and ( logdate <= 12/31/2003 )
LOGS     6 where type = "P" and ( logdate >= 01/01/2004 and logdate <= 06/30/2004 )
LOGS     7 where type = "P" and ( logdate >= 07/01/2004 and logdate <= 12/31/2004 )
LOGS     8 where type = "P" and ( logdate >= 01/01/2005 and logdate <= 06/30/2005 )
LOGS     9 where type = "P" and ( logdate >= 07/01/2005 and logdate <= 12/31/2005 )
LOGS    10 where type = "P" and ( logdate >= 01/01/2006 )
LOGS    14 where type = "L" and ( logdate <= 08/01/2002 )
LOGS    15 where type = "L" and ( logdate  > 08/01/2002 )
LOGS    16 where type = "M"
LOGS    17 where type = "U"
```

The Code

- **start.p**
- **dlctl.p**
- **dlclear.p**
- **dlwrap.p**
- **dlx.p**

start.p

```
/* start.p
 *
 */

run ./dotp/dlclear.p.

pause 1 no-message.
run ./dotp/dlctl.p.

pause 1 no-message.
run ./dotp/dlmon.p.

return.
```

dlclear.p

```
/* dlclear.p
 *
 */

define new global shared variable
 dlstart as character no-undo format "x(20)".

dlstart = "".

for each dlctl exclusive-lock:
  assign
    dlctl.dumped    = 0
    dlctl.loaded    = 0
    dlctl.err_count = 0
    dlctl.note      = ""
    dlctl.xstatus   = ""
  .
end.

return.
```

dlctl.p

```
/* dlctl.p
 *
 */

for each dlctl no-lock where dlctl.active = yes
      by dlctl.expected descending:

  os-command silent
    value(
      "mbpro -pf dl.pf -p ./dotp/dlwrap.p -param " + '"' +
        string( dlctl.thread ) + "|" +
        dlctl.tblname + "|" +
        dlctl.criteria +"|" +
        dlctl.pname + '"' +
      " >> /tmp/dl/error.log 2>&1 ~&"
    ).

end.

return.
```

dlwrap.p

```
/* dlwrap.p
 *
 */

define variable thr as character no-undo.
define variable tbl as character no-undo.
define variable cri as character no-undo.
define variable pnm as character no-undo.

thr = entry( 1, session:parameter, "|" ).
tbl = entry( 2, session:parameter, "|" ).
cri = entry( 3, session:parameter, "|" ).
pnm = entry( 4, session:parameter, "|" ).

if pnm = "" then pnm = "dlx".

pnm = "./dotp/" + pnm + ".p".

run value( pnm ) value( thr ) value( tbl ) value( cri ).

return.
```

dlx.p - part 1

```
/* {1} = thread number
 * {2} = table name
 * {3} = WHERE clause */

define variable flgname as char no-undo initial /tmp/dl/{2}.{1}.flg".
define variable logname as char no-undo initial "/tmp/dl/{2}.{1}.log".

define stream unloaded.
output stream unloaded to value( "/tmp/dl/{2}.{1}.d" ).

output to value( logname ) unbuffered.
put today " " string( time, "hh:mm:ss" ) " {2} {1}" skip.

disable triggers for dump of src.{2}.
disable triggers for load of dst.{2}.


define query q for src.{2}.
open query q for each src.{2} no-lock {3}.
```

dlx.p - part 2

```
load_loop: do for dlctl, dst.{2} while true transaction:
  dump_loop: do while true:

    get next q no-lock.
    if not available src.{2} then leave load_loop. /* end of table */
    create dst.{2}.
    buffer-copy src.{2} to dst.{2} no-error.
    d = d + 1.
    if error-status:num-messages = 0 then
      l = l + 1.
     else
      do:
        delete dst.{2}.
        e = e + 1.
        export stream unloaded src.{2}.
        find dl.dlctl exclusive-lock
             where dlctl.tblname = "{2}" and dlctl.thread = {1}.
        dlctl.err_count = e.
        next dump_loop.
      end.
```

dlx.p - part 3

```
if d modulo 100 = 0 then
     do:
       find dl.dlctl exclusive-lock
            where dlctl.tblname = "{2}" and dlctl.thread = {1}.
       assign
         dlctl.dumped = d
         dlctl.loaded = l
       .
       file-info:file-name = flgname.
       if file-info:full-pathname = ? then
         do:
           dlctl.note = "stopped".
           leave dump_loop.
         end.
       leave dump_loop.
     end.
  end.   /* dump_loop    */
end.     /* load_loop    */
```

dlx.p - part 4

```
do for dl.dlctl transaction:

  find dl.dlctl exclusive-lock
       where dlctl.tblname = "{2}" and dlctl.thread = {1}.
  assign
    dlctl.dumped    = d
    dlctl.loaded    = l
    dlctl.err_count = e
    dlctl.xstatus   = "done"
    dlctl.note      =
      ( if dlctl.note <> "stopped" then "complete" else "stopped" )
  .
  if dlctl.note <> "stopped" then
    do:
      if d < dlctl.expected then dlctl.note = "short".
      if d > dlctl.expected then dlctl.note = "long".
    end.

  if e > 0 then dlctl.note = dlctl.note + "+errors".

end.
```

Validation

- **Belt & Braces!!!**
  - **Compare Record Counts**
    - **dlcompare.p (for binary d&l)**
  - **Check Logs for Known & Unknown Errors**
    - `grep -i -f error.list`
    - **fail, error, (1124)**
  - **Check for Success**
    - `grep " 0 errors" /tmp/dl/*.log | wc -l`
    - `ls -l /tmp/dl/*.d`

Recap

- **Good Reasons to Dump & Load**
  - **Move to a new platform.**
  - **Reconfigure a storage decision.**
  - **Take advantage of a hot new db features!**
- **How To Get There Safely…**
- **… and Quickly!**

Questions

http://www.greenfieldtech.com/downloads.shtml